1987

Determination of network attributes from a high resolution terrain data base

Choi, Seok Cheol

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DETERMINATION OF NETWORK ATTRIBUTES
FROM
A HIGH RESOLUTION TERRAIN DATA BASE

by

Seok Cheol Choi

September 1987

Thesis Advisor Samuel H. Parry

ECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; Distribution is unlimited.
b DECLASSIFICATION/DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 55	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
c ADDRESS (City, State and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

1 TITLE (Include Security Classification): DETERMINATION OF NETWORK ATTRIBUTES FROM A HIGH RESOLUTION TERRAIN DATA BASE

2 PERSONAL AUTHOR(S): CHOI, Seok Cheol

13a TYPE OF REPORT	13b TIME COVERED FROM ____ TO ____	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis		1987 September	98

6 SUPPLEMENTARY NOTATION

7 COSATI CODES: FIELD	GROUP	SUB-GROUP	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
			Determination of network attributes, minimum travel time path, minimum distance path

9 ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this research is to develop algorithms for mapping terrain characteristics from a 100 meter grid representation to arcs and nodes of a transportation network. The algorithms capture elevation and trafficability parameters as they relate to both the arc itself and to the off-arc characteristics. These network parameters are directly usable in appropriate optimization algorithms to determine minimum travel time path and minimum distance path through the network for a variety of maneuver unit types and missions.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL: Prof. Samuel H. Parry	22b TELEPHONE (Include Area Code): 408-646-2779; 22c OFFICE SYMBOL: 55Py

D FORM 1473, 84 MAR — 83 APR edition may be used until exhausted; All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

Determination of Network Attributes from A High Resolution Terrain Data Base

by

Seok Cheol Choi
Major, Republic of Korea Army
B.A., Korea Military Academy, 1979

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

ABSTRACT

The purpose of this research is to develop algorithms for mapping terrain characteristics from a 100 meter grid representation to arcs and nodes of a transportation network. The algorithms capture elevation and trafficability parameters as they relate to both the arc itself and to the off-arc characteristics. These network parameters are directly usable in appropriate optimization algorithms to determine minimum travel time path and minimum distance path through the network for a variety of maneuver unit types and missions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. PURPOSE AND GOALS

The purpose of this research is to develop algorithms for mapping terrain characteristics from a 100 meter grid representation to arcs and nodes of a transportation network. The algorithm must capture elevation and trafficability parameters as they relate to both the arc itself and to the off-arc characteristics. These network parameters must be directly usable in appropriate optimization algorithms to determine minimum travel time path and minimum distance path through the network for a variety of maneuver unit types and missions.

After reviewing the literature and discussing the problem with several terrain model experts, it was determined that no automated procedure for accomplishing this mapping exists. Therefore the procedures and associated algorithms presented in this thesis represent original research and development.

B. AIR LAND RESEARCH MODEL METHODOLOGIES

The purpose of the Air Land Advanced Research Model (ALARM) is to develop a systemic corps level combat model, that is, one that operates without human intervention. This idea is not new, but ALARM takes a different approach than previous efforts to simulate the military decision process.

In most combat models, the structures for planning and for execution are included in the same program. In ALARM the planning function and the execution function will be separate programs. The Planning model will be a carefully designed collection of programs that provide orders to the Execution model at the appropriate times.

A second major difference between the ALARM design and existing models is in the method used to make decisions. All of the models reviewed when developing the ALARM strategy depended either on threshold parameters or on a series of "IF THEN" decision rules to make decisions.

Combat models using a threshold parameter strategy usually require a large number of threshold parameters or groups of parameters. In such models a threshold parameter (or group) is needed for every possible decision that any unit represented in the combat simulation might have to make. This leads to numerous parameters and

thresholds. Most parameters of this type have no reasonable real world counterparts. In other words, parameters used for thresholding are frequently contrived and based on unexplainable and undefendable measures. This makes it difficult to create data bases for such models, and difficult to place any confidence in the analytical results from the models.

Models that use a series of decision rules are lacking in that they require a very limited view of the decision process. A decision can be made only if a series of decision rules to support the decision exist within the software that is developed for the model, and if those specific rules are satisfied. When a situation is encountered for which there is no specific decision rule, a default rule must be implemented. These default rules are frequently inappropriate. The more complicated the decision rules become, the better the model usually is, but the more difficult it becomes to maintain the model and the harder it is to analyze results from the model. The cause and effect relationships established when running the model are usually closely tied to the decision rules. Thus the output or results from such a simulation are built into the model. The foregoing discussion emphasizes problems with the methods currently accepted and used to model the decision process. These methods are recognized because they have their uses, applications and foundations in the human decision making process. Human beings do, on some level, use the threshold initiated decision rule process in their decision making procedures. These concepts are usually used to start or constrain the decision process and not to make the decision itself.

The premise of ALARM is to use both threshold strategy and decision rules, but not within the same framework as other models to date. Thresholds will be used to determine when the planning or decision making procedures should be executed. Decision rules will be used to limit alternatives. They will not be used explicitly to make combat decisions. ALARM will use network based methodologies to itemize alternatives so that a decision can be made. It is believed that this methodology is closer to what actually happens in the human decision making process.

ALARM decision processes, called Decision Tasks, will use three unique methodologies that are the mainstay of the ALARM concept and make it different than other current research. These methodologies are time domain networks, cartesian space networks, and the Generalized Value System. Each methodology is briefly described below.

1. **Time Domain Networks**

Time domain networks are designed to handle the planning function or activity within the ALARM system. A time domain network consists of nodes and arcs (or links) connecting the nodes. The key to time domain networks is that arcs do not represent distance, but represent the passage of time or the completion of a sub-activity which leads to the completion of the entire activity represented by the network.

The time domain network is used to develop high level mission requirements for all subordinate units. It is similar to a PERT or CPM network in that a sub-activity cannot be started until all merging activities are completed. The input to the activity network is a desired activity completion time and an acceptable friendly force attrition level, and may include a required level of attrition to enemy forces.

2. **Cartesian Space Networks**

The Cartesian space network is a series of networks each representing a different aspect of the battlefield. Each network represents physical connections between points on the battlefield. ALARM will have three or more Cartesian space networks. The three networks that have been identified are terrain and transportation networks, communication networks, and logistics resupply networks. This thesis deals explicitly with the terrain and transportation networks.

3. **Generalized Value System**

The generalized value System (GVS) is a procedure for quantifying the capabilities and importance of entities on the battlefield at some future time (t+x). It does this by developing algorithms to predict future entity or situation states at time (t+x) based on the situation at time (t). Thus GVS provides the framework for forecasting future states of entities in continuous time.

Various combinations of exponential functions are used to represent these forecasts. For example, the power and value of non-combat entities such as bridges and road interdiction points; combat support units; service support units; and combat units can all be determined from a continuous function with respect to time using the various exponential algorithms. These values can then be used to determine the specific place or target against which to plan an air strike or a ground attack at time (t+x). Thus different targets can be selected by the Decision Task depending on the time at which strike assets are available. The decision will be based not on the current value of the target, but on its predicted future value in relation to the rest of the Air Land battle. To do this, all targets and units must be assigned consistent values so that

comparisons can be made across the diverse set of targets available on the Air Land battlefield. The common bond among these varying targets is a value system that measures the target's value in terms of its usefulness to combatants. Thus the value of a bridge is not measured simply by its width or length, but is tied to the number of enemy or friendly units that could cross the bridge to maneuver into favorable combat positions. As time passes, the bridge's value could increase based on heavy combat areas and then could decrease when all relevant units have successfully crossed the bridge.

C. SCOPE OF THE THESIS

In attempting to achieve the goals of this thesis, various terrain modelling techniques will be described in Chapter II as a motivation for the network algorithms developed. The algorithms for a single arc will form the basis of the total model and will be presented in Chapter III. The full network implementation will be presented in Chapter IV to determine the minimum time and minimum distance paths on an undirected graph. Chapter V will discuss the results of several runs of the model to illustrate its various capabilities. Finally, Chapter VI will provide recommendations for continued enhancements of the model.

II. BACKGROUND

A. TERRAIN MODELLING TECHNIQUES

Representing combat functions as a system of networks is a departure from the current practice of explicit terrain representation. Current modelling techniques require large data bases to represent numerous aspects of terrain such as slope, elevation, forestation, and cities. The network approach is an attempt to represent the terrain, primarily through use of the transportation system, as an abstraction of its actual state. The topology characteristics required by explicit terrain models will be reflected in the arc and node attributes.

The research model will gain an economy of representation of the topology because it has the flexibility to represent only those terrain features applicable to a stated level of resolution. The corps rear is sparsely populated with combat and support units relative to the overall size of the corps sector. Furthermore, large portions of the corps sector will never support unit movement or facilities and have no reason to be modelled. A network representation allows one to select only those features which, because of their nature, give the controlling force a marked advantage over the other force (e.g. key terrain). Additionally, those roads or cross-country arcs which can either expedite or delay movement can be identified and modelled.

In a corps level model, direct fire combat requires observation, detection, and lines of sight along a narrow strip of terrain relative to the overall size of the sectors of the two forces. Existing methods of terrain representation may still be required to model the details of such combat. The majority of units moving in a corps sector, especially the CS/CSS assets, do so without receiving direct fire. The movement of the units can be modelled via a network abstraction of the transportation system. Numerous algorithms exist which allow one to exploit the features and structure of such a mathematical model.

Many of the terrain modelling techniques in current use restrict the model to one level of resolution and force the terrain data base to store attributes for large sections of the battlefield which are not used. To put current terrain modelling procedures in perspective, the following is a short discussion of three terrain modelling methods: hex terrain (used in CORDIVEM), digital terrain (DYNTACS) and network model.

B. HEX TERRAIN

The Corps-Division Evaluation Model (CORDIVEM) is a corps level model in which the terrain is represented as a series of hexagons, (hexes), each approximately 3.5 kilometers in diameter. The terrain underlying each hex is characterized by a three digit array which represents urbanization, forestation, and roughness. Each individual attribute is an average of the Defence Mapping Agency's data base of points 12.5 meters apart which underlie each CORDIVEM hex. The attribute based on the resulting average is then assigned a code in CORDIVEM and assumed homogeneous throughout the hex [Ref. 1: p. 16]. Figure 2.1 illustrates the codes for each hex attribute.

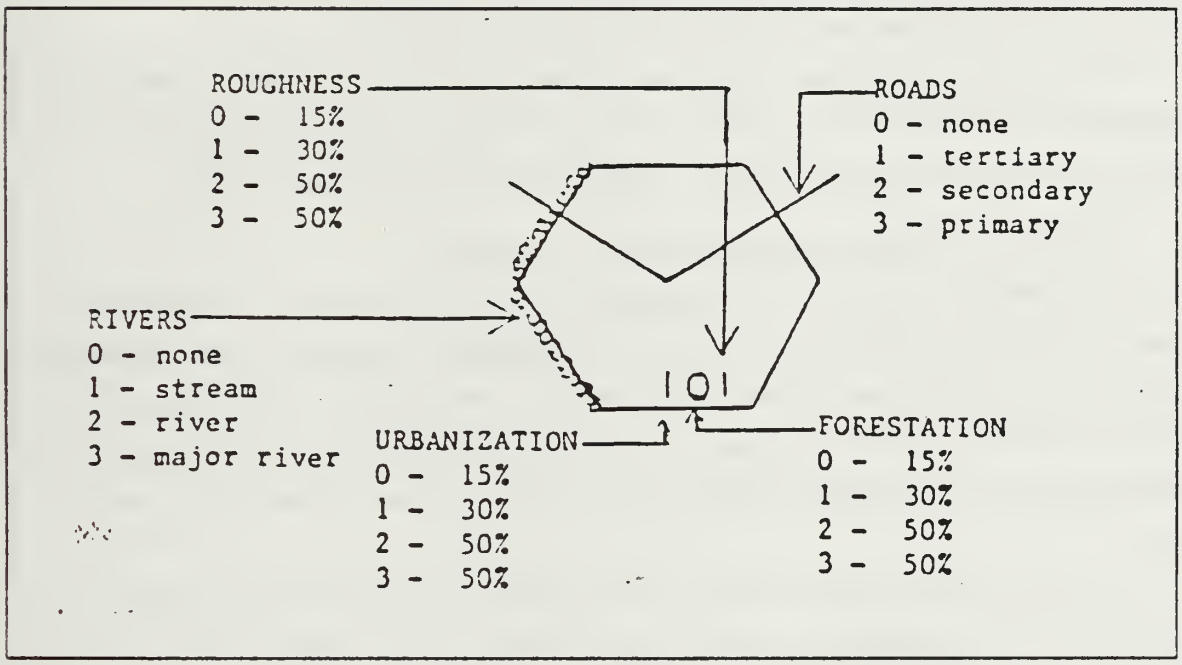

Figure 2.1 CORDIVEM Terrain Features.

The geometry of the hex is used to represent unit movement, roads, rivers and obstacles. Unit movement and a road system, if one exists, are modelled from the center of one hex to the center of an adjacent hex. Movement on a cross-country route is penalized by reducing a unit's allowable speed. Rivers are modelled along the edge of a hex, and coded as indicated in Figure 2.1. An implicit bridge is assumed at the intersection of all roads and rivers. The bridge can be explicitly modelled if it may affect movement factors or possibly be destroyed. An obstacle occurs on the edge of

the hex and is explicitly inserted into or removed from the model. All obstacles are created before the execution of the model; the dynamic construction and reduction of obstacles are not modelled.

C. DIGITIZED TERRAIN

The Dynamic Tactical Simulation (DYNTACS) model is a two-sided, dynamic Monte Carlo simulation capable of modelling the combat process from the individual crew to battalion engagements which utilizes a concept referred to as digitized terrain [Ref. 2: p. 9]. Macro terrain in DYNTACS is usually represented as 100 meter squares with attributes for the corners being supplied by data available from the Waterways Experiment Station. These corner attributes provide information about such features as elevation, forestation, and location. Each square is then divided diagonally, resulting in a series of equal sized triangles, varying in slope and orientation.

This approach assumes that the terrain modelled by each triangle is homogeneous and that elevation changes spacings. Sudden changes in the terrain or its surface are not detected; instead, they are represented with geometric overlays.

D. NETWORK MODEL FOR MOBILITY IN ALARM

The Cartesian space network compacts the terrain database still further by limiting battlefield movement to a subset of battlefield locations. The allowable locations are represented as a network overlayed on the battlefield map. Nodes of the network represent physical locations on the map. Arcs of the network represent possible movement paths between nodes such as roads, trails, or likely cross country routes.

Each node and arc of the network has mobility attributes which determine the limiting speed of combatants moving through that node or along that arc. The attributes which are stored in the network representation can be either terrain features (which will support an explicit mobility model) or mobility multipliers (for a simplified implicit model).

Added flexibility can be gained by storing several sets of mobility attributes for some arcs. The main attribute set might represent movement along a road, while a second attribute set could represent cross country mobility in the near vicinity of the road if the road itself cannot be used.

The main problem with network models for battlefield movement is that movement is restricted to the network, and this may limit tactical flexibility.

Preparation of the network must be carefully coordinated with the tactical scenario to ensure sufficient movement opportunities. This limitation would seem to be most severe in the direct fire battle zone where cross country maneuver is common. Areas away from the front lines can more readily be represented by a network model. A major advantage of the network model is that it allows efficient network optimization algorithms to be applied to the problem of route selection for moving combatants.

To date, data for network attributes have been derived manually using map analysis. This technique is both extremely manpower intensive and inaccurate. The requirement for an automated process to generate network attributes directly from digitized terrain data bases serves as the motivation for this research. The algorithms to determine attributes for a single arc are described in the next chapter.

III. SINGLE ARC METHODOLOGY

A. OBJECTIVE AND PROBLEM DEFINITION

The objective of this research is to develop algorithms for mapping terrain characteristics from a 100 meter grid representation to arcs and nodes of a transportation network. The algorithm must capture elevation and trafficability parameters as they relate to both the arc itself and to the off-arc characteristics. These network parameters must be directly usable in appropriate optimization algorithms to determine minimum travel time path and minimum distance path through the network for a variety of maneuver unit types and missions.

Methodology for the single arc attributes which is used for the Cartesian space network is presented in this chapter and is related to the computer program for the single arc attributes in Appendix C.

B. MODEL DATA FILE

The program used four disk data files, one for 100 meter grid square data, one for the node characteristics, one for the arc characteristics which represents the networks overlayed on the gridded terrain, and one for the speed data related to maneuvering units.

1. 100 Meter Grid Square Data

The grid square data represents the altitude and characteristics of each point spaced 100 meters apart on the terrain. The source of the data is the Combined Arms Center at Fort Leavenworth, which is the same data base originally used for CORDIVEM. The gridded terrain is used to determine the attributes of networks overlayed on the terrain. A typical network overlayed on the digitized terrain is shown in Figure 3.1.

Two characteristics are used to describe a point of gridded terrain: three digit altitude and an integer surface feature type. The altitude is the height from sea level or simple map altitude. The characteristics of the surface feature type in the network are used for determining the arc width for maneuver along the arc. Table 1 shows how data for the 100 meter resolution surface feature types were encoded.

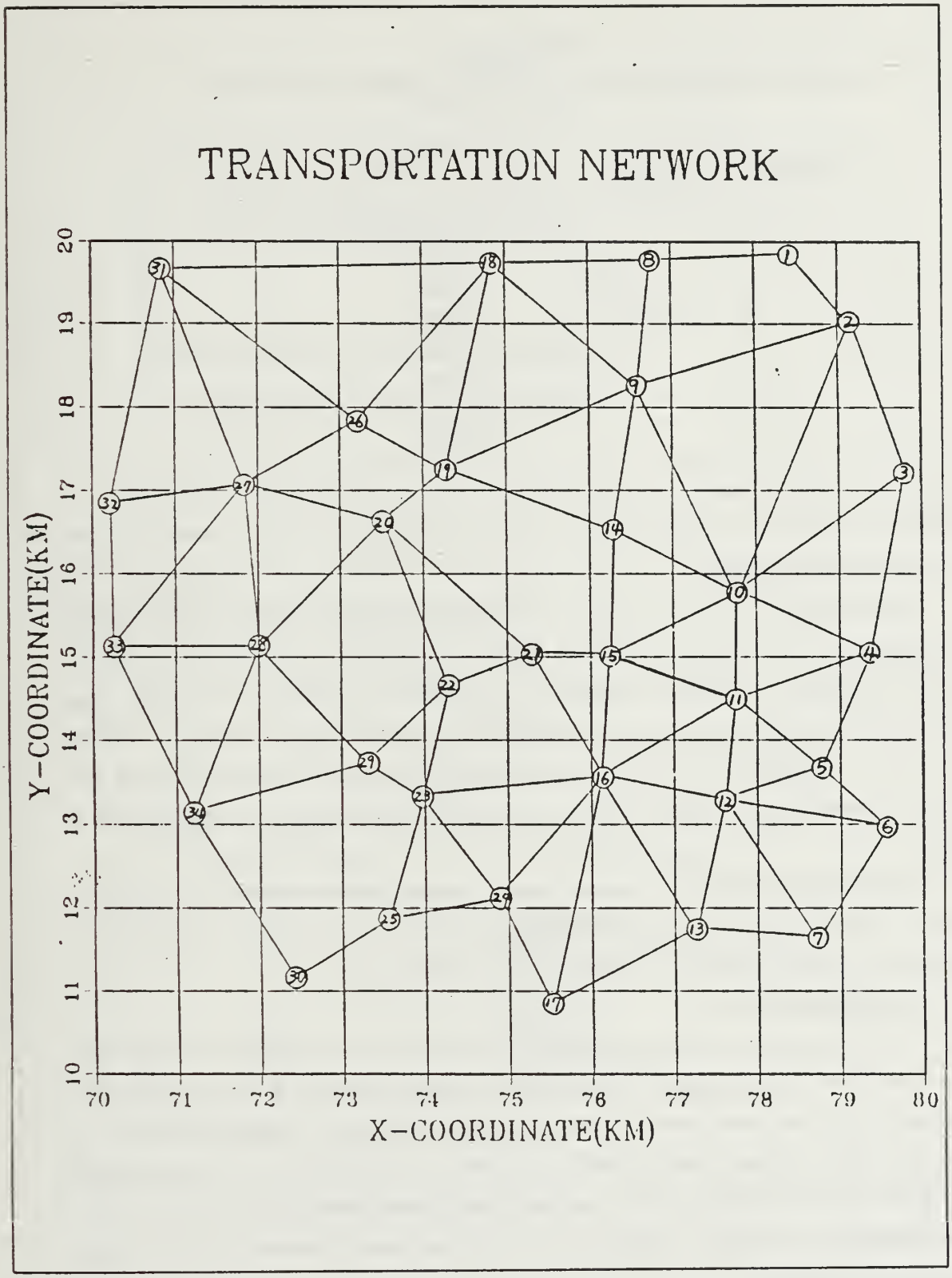

Figure 3.1 Transportation Network.

TABLE 1
CHARACTERISTICS OF SURFACE FEATURE TYPES

Integer Value	Surface Feature
0	No Elevation/No feature data
1	Forest
2	Urban
3	Marsh
4	Null(Elevation/No feature data)
5	Water
6	Heath(Waste land with shrubs)
7	Open

2. **Node Characteristics**

In the network, nodes are used to represent actual terrain features on a map. Four characteristics are used to describe a node: an integer node identification number, three digit latitudinal coordinate, three digit longitudinal coordinate, and node type. The coordinates are simple map grid coordinates. The node type is entered as a single integer value. Eventually, each node has an altitude linearly interpolated from 100 meter grid square data. Table 2 shows the integer values and the terrain feature it represents.

The primary function of the node in the transportation network is to relate the junction of arcs in the data representation to physical locations on the map. In addition, they represent possible objectives in a combat mission.

3. **Arc Characteristics**

The arcs in the network represent feasible routes of maneuver from one location (node) to another. For a route to be considered feasible, it must provide for the maneuver of at least one column of tracked vehicles, wheeled vehicles, or dismounted troops. Three characteristics are used to represent an arc: head node identification, tail node identification, and arc type. These characteristics provide data for the algorithms to compute appropriate arc traversal times, distances and flow rates from node A to node B.

TABLE 2
TYPE OF NODE

Integer Value	Terrain Feature
1	City
2	Village
3	Road Junction
4	Hill Top
5	Other

The head node identification is the identification number of the node where the arc originates. The tail node identification is the identification number of the node where the arc terminates. The characteristics of the arc is an integer code for one of seven possible arc types as shown in Table 3.

TABLE 3
TYPE OF ARC

Integer Value	Terrain Feature
1	Autobahn
2	Concrete or Asphalt Road
3	Dirt Road
4	Railroad
5	Forest
6	Open Country
7	Bridge or Tunnel

4. **Speed Data**

The speed for unit maneuver is a function of arc type, unit type and unit formation. The possible speed represents the speed on the specific arc for the specific unit for each unit formation. It is assumed that the formation width of a single column is less than the width of the road itself. If the width of the unit formation is less than the width of the road itself, then arc speed is on-road speed. Otherwise, arc speed is off-road speed, because on-road speed is greater than off-road speed with same arc type and unit type.

C. STEPS IN PROCESS

To translate 100 meter grid square data to attributes of arcs, the network structure is used. The network structure can exploit the mathematical nature of the variables of the decision process by representing these variables as characteristics of the arcs and nodes in the network overlayed on the gridded terrain. It is an objective in developing this network to adequately describe the terrain and combat effects on maneuverability necessary for the implementation. The process of translation and implementation from 100 meter grid square data to attributes of arcs is described for single arc geometry and later for multiple arc geometry and flow rates in the network.

1. **Single Arc Attributes**

a. Geometry

Consider an arc which links node A to node B. To represent the coordinates of a node in three dimensions, the Cartesian coordinate (x, y, z) system is used. Each node has an altitude interpolated from the 100 meter grid square data. From the coordinates of node A and node B, the total distance between node A and node B, and the angle of inclination of the arc is computed. The angle of inclination is determined by the slope between the x-axis and y-axis. The definition of angle of inclination of a line that crosses the x-axis is the smallest positive angle that the line makes with the positively directed x-axis [Ref. 3: p. 50].

If the angle of inclination is between 45 degrees and 135 degrees, then the x-axis is used as reference axis for calculation of distance between each point along the arc. If the angle of inclination is greater than 135 degrees or less than 45 degrees, then the y-axis is used as the reference axis. Using this angle, coordinates of the cross points between a point on the arc and a point on the each 100 meter grid line for the reference axis is computed. After determining the coordinates of the points along the

arc, the distance between each pair of points is calculated step by step until the cross point reaches the tail node based on the distance formula for the plane. Figure 3.2 shows an example of crossing points between the arc (which is Node 1 to Node 2) and the x-axis 100 meter grid line along the arc.

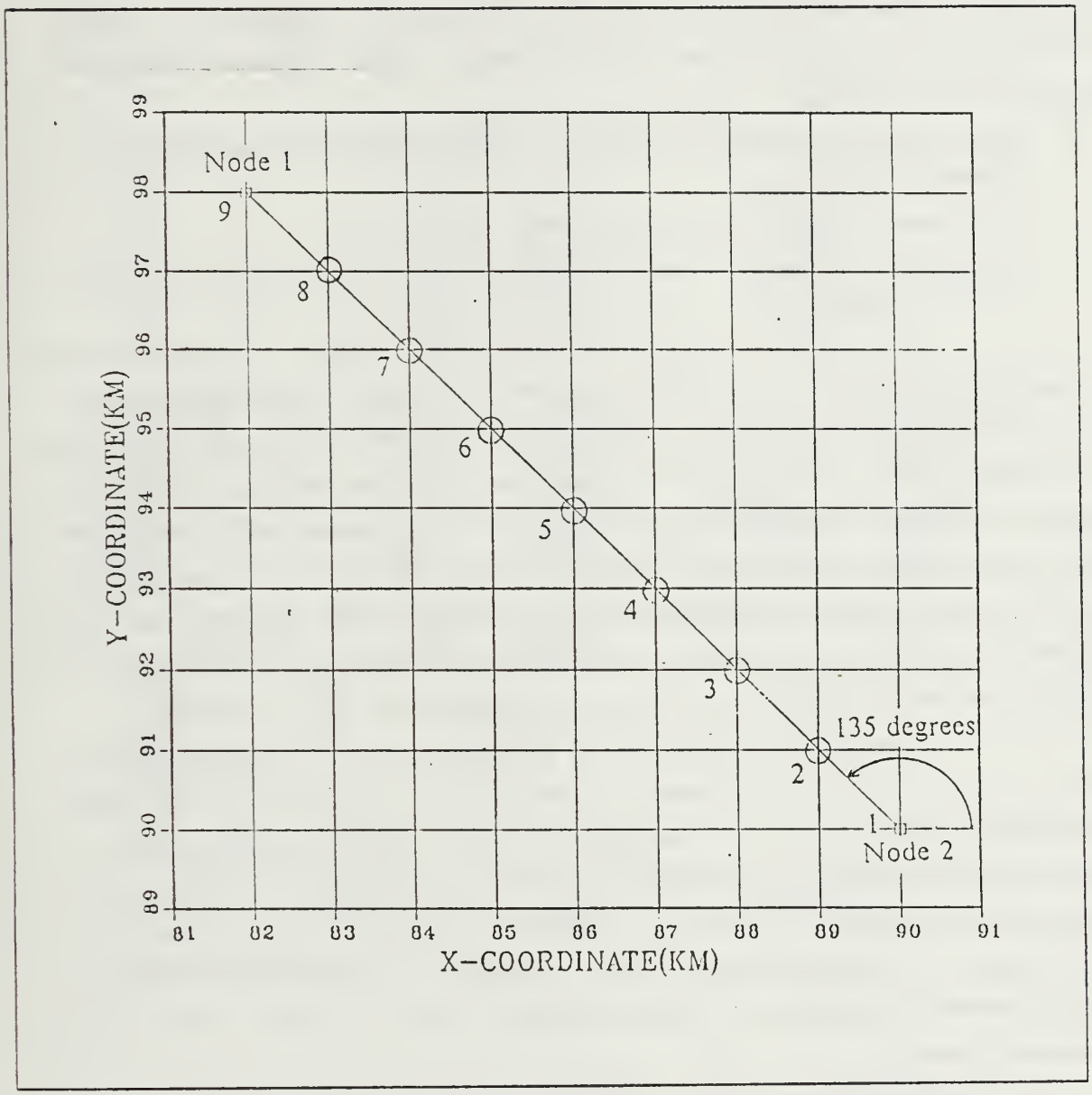

Figure 3.2 Crossing Points.

b. Slope

(1) *Along The Arc.* To determine the gradient pattern along the arc, the slope of the terrain between each pair of points along the arc is calculated. Coordinates and altitudes of each point along the arc and the difference of altitudes between each sequential pair of points are used to calculate the slopes between each pair of points along the arc. The coordinates of each point along the arc are computed by using the the point-slope equation of the line. As a measures of terrain steepness along the arc, three factors are computed:

- Slope Change - sum of the absolute value of slopes between each pair of adjacent sub-arcs.
- Total Slope Change - sum of slope change along the arc.
- Average Slope Change - total slope change divided by the number of pair of sub-arcs along the arc.

The slope of the surface along the arc is related to the feasible routes of maneuver for each type of unit from one node to another. Some arcs provide for the maneuver of dismounted troops, but tracked and wheeled vehicle movement may be infeasible. Some arcs provide for the movement of tracked vehicles to climb steep slopes but wheeled vehicle movement is not possible. The maneuverability on a given slope depends on unit type (tracked vehicles, wheeled vehicles or dismounted troops) and soil strength which is measured by rating cone index points: the higher the index numbers, the greater the soil strength. The maximum negotiable slope can be used after obtaining soil strength data for arcs as shown in Appendix G. [Ref. 4: p. 5-19]

Table 4 shows an example of general information and slopes along the arc from Node 1 to Node 2 as shown in Figure 3.2. XM represents the slope between starting node and terminal node from the coordinates of the x, y plane on the arc. ANGLE represents the angle of inclination of the arc. ACT.DIST is actual distance between two nodes along the surface, but DISTANCE is the distance on the plane or on the map. It is assumed that the distance of the arc is DISTANCE. AVERAGE SLOPE represents the difference between altitude of Node 1 and Node 2 divided by the distance of the arc.

(2) *Perpendicular to Arc.* After determining the slope of the terrain along the arc, the slopes of the terrain along lines perpendicular to the arc are computed. The starting point is the cross point between the x-axis grid line and the arc, if the angle of inclination is greater than 45 degrees and less than 135 degrees. Otherwise, the starting point is the cross point between the y-axis grid line and the arc.

TABLE 4

SLOPE AND DISTANCE

NODE 2 X	NODE 2 Y	NODE 1 X	NODE 1 Y	XM	ANGLE	DISTANCE	ACT. DIST
9000	9000	8200	9800	-1.00	135.00	1131.37	1131.95

POSITION	POINTS ALONG ARC COORDINATE(X : Y)	ALTITUDE
POINT 1	(9000.0 9000.0)	367.0
POINT 2	(8900.0 9100.0)	367.0
POINT 3	(8800.0 9200.0)	367.0
POINT 4	(8700.0 9300.0)	372.0
POINT 5	(8600.0 9400.0)	371.0
POINT 6	(8500.0 9500.0)	367.0
POINT 7	(8400.0 9600.0)	361.0
POINT 8	(8300.0 9700.0)	354.0
POINT 9	(8200.0 9800.0)	348.0

POSITION	DELTA H	DISTANCE	SLOPE	ACTUAL DIST
POINT 1 - 2	0.0	141.4	0.000	141.4
POINT 2 - 3	0.0	141.4	0.000	141.4
POINT 3 - 4	5.0	141.4	0.035	141.5
POINT 4 - 5	-1.0	141.4	-0.007	141.4
POINT 5 - 6	-4.0	141.4	-0.028	141.5
POINT 6 - 7	-6.0	141.4	-0.042	141.5
POINT 7 - 8	-7.0	141.4	-0.049	141.6
POINT 8 - 9	-6.0	141.4	-0.042	141.5

POSITION	SLOPE CHANGE
POINT 1 - 2	0.00
POINT 2 - 3	0.04
POINT 3 - 4	0.04
POINT 4 - 5	0.02
POINT 5 - 6	0.01
POINT 6 - 7	0.01
POINT 7 - 8	0.01

TOTAL SLOPE CHANGE : 0.1697

AVERAGE SLOPE CHANGE : 0.0212

NODE 2:	ALTITUDE 1	NODE 1:	ALTITUDE 2
	367.0		348.0

AVERAGE SLOPE : -0.0168

The altitude of the starting point is calculated by linear interpolation from the nearest left intersection between the x-axis and y-axis to the nearest right intersection between the x-axis and y-axis line. In the following discussion, it is assumed that the angle is 135 degrees. Table 5 and Figure 3.3 are used to illustrate the following discussion.

The sequence of calculations is to determine the slope of left off-road and right off-road along the perpendiculars. On the left side the first distance is from the starting point to the first cross point between the perpendicular line and y-axis grid line. The second distance is from the same starting point as the first one to the second cross point, etc. The altitude of the cross point between the perpendicular line to the arc and the y-axis grid line is computed by the same interpolation procedure as for the starting point. Using the first distance and altitude of the starting point and the first cross point between the perpendicular line and y-axis grid line, the first slope is calculated. The process is continued until terminated by the stopping rule described later. After computing each slope on the perpendicular line for the initial point (which is the head node), slopes along the perpendicular for subsequent points along the arc are computed, terminating with the tail node. After determining the slope on the left side, the right side slopes are computed using the same procedures. Before calculating the slope on the perpendicular line to the arc, coordinates of each cross point between the perpendicular line to the arc and the y-axis grid line are determined. The coordinates of each point on the perpendicular line to the arc are computed using the same point-slope equation for the coordinates along the arc. Also, the difference in altitude of each pair of points is computed.

The slopes on the perpendicular lines are related to the feasible arc width for movement from left boundary to right boundary. The arc width measured along the perpendicular line to the arc is considered for the trafficable path for each specific type of unit. Table 5 shows an example of slopes used to determine the trafficable width of an arc for the movement of vehicle units. For example, POINT 4 - 1 represents the first (left) cross point between the y-axis 100 meter grid line and the line perpendicular to the arc at the fourth point from the starting node. POINT 4 - 5 is the boundary point to left of the arc, POINT 4 - 6 is the first point to the right, and POINT 4 - 11 is the boundary point to right of the arc as shown in Figure 3.3. SLOPE represents DELTA H divided by DISTANCE.

(3) *Arc Width.* Two types of stopping rules are used for determining the arc width: a code change rule and a slope change rule. Briefly stated, an arc width boundary is established along a line perpendicular to an arc when the terrain code changes from type i to type j as described later. Also, a boundary may be established because the slope change along the perpendicular exceeds an input threshold value.

TABLE 5

SLOPE ON PERPENDICULAR LINE TO ARC

SOME POINTS OFF ARC (FROM EACH POINT ON THE ARC)

COORDINATE (X : Y)			ALTITUDE	DELTA H	DISTANCE	SLOPE
POINT	1- 1	(8900 8900)	374.00	7.0	141.42	0.05
POINT	1- 2	(9100 9100)	364.00	-3.0	141.42	-0.02
POINT	2- 1	(8800 9000)	374.00	7.0	141.42	0.05
POINT	2- 2	(9000 9200)	203.00	-164.0	141.42	-1.16
POINT	3- 1	(8700 9100)	374.00	7.0	141.42	0.05
POINT	3- 2	(8900 9300)	366.00	-1.0	141.42	-0.01
POINT	4- 1	(8600 9200)	371.00	-1.0	141.42	-0.01
POINT	4- 2	(8500 9100)	384.00	12.0	282.84	0.04
POINT	4- 3	(8400 9000)	404.00	32.0	424.26	0.08
POINT	4- 4	(8300 8900)	443.00	71.0	565.69	0.13
POINT	4- 5	(8200 8800)	499.00	127.0	707.11	0.18
POINT	4- 6	(8800 9400)	367.00	-5.0	141.42	-0.04
POINT	4- 7	(8900 9500)	358.00	-14.0	282.84	-0.05
POINT	4- 8	(9000 9600)	354.00	-18.0	424.26	-0.04
POINT	4- 9	(9100 9700)	351.00	-21.0	565.69	-0.04
POINT	4-10	(9200 9800)	348.00	-24.0	707.11	-0.03
POINT	4-11	(9300 9900)	348.00	-24.0	848.53	-0.03
POINT	5- 1	(8500 9300)	380.00	9.0	141.42	0.06
POINT	5- 2	(8400 9200)	381.00	10.0	282.84	0.04
POINT	5- 3	(8300 9100)	400.00	29.0	424.26	0.07
POINT	5- 4	(8200 9000)	440.00	69.0	565.69	0.12
POINT	5- 5	(8100 8900)	482.00	111.0	707.11	0.16
POINT	5- 6	(8000 8800)	499.00	128.0	848.53	0.15
POINT	5- 7	(7900 8700)	499.00	128.0	989.95	0.13
POINT	5- 8	(7800 8600)	492.00	121.0	1131.37	0.11
POINT	5- 9	(8700 9500)	361.00	-10.0	141.42	-0.07
POINT	5-10	(8800 9600)	354.00	-17.0	282.84	-0.06
POINT	5-11	(8900 9700)	351.00	-20.0	424.26	-0.05
POINT	5-12	(9000 9800)	351.00	-20.0	565.69	-0.04
POINT	5-13	(9100 9900)	348.00	-23.0	707.11	-0.03
POINT	6- 1	(8400 9400)	374.00	7.0	141.42	0.05
POINT	6- 2	(8300 9300)	394.00	27.0	282.84	0.10
POINT	6- 3	(8200 9200)	398.00	31.0	424.26	0.07
POINT	6- 4	(8100 9100)	420.00	53.0	565.69	0.09
POINT	6- 5	(8000 9000)	453.00	86.0	707.11	0.12
POINT	6- 6	(7900 8900)	466.00	99.0	848.53	0.12
POINT	6- 7	(7800 8800)	469.00	102.0	989.95	0.10
POINT	6- 8	(7700 8700)	469.00	102.0	1131.37	0.09
POINT	6- 9	(8600 9600)	358.00	-9.0	141.42	-0.06
POINT	6-10	(8700 9700)	351.00	-16.0	282.84	-0.06
POINT	6-11	(8800 9800)	348.00	-19.0	424.26	-0.04
POINT	6-12	(8900 9900)	348.00	-19.0	565.69	-0.03
POINT	7- 1	(8300 9500)	367.00	6.0	141.42	0.04
POINT	7- 2	(8200 9400)	381.00	20.0	282.84	0.07
POINT	7- 3	(8100 9300)	384.00	23.0	424.26	0.05
POINT	7- 4	(8000 9200)	400.00	39.0	565.69	0.07
POINT	7- 5	(7900 9100)	404.00	43.0	707.11	0.06
POINT	7- 6	(7800 9000)	407.00	46.0	848.53	0.05
POINT	7- 7	(7700 8900)	417.00	56.0	989.95	0.06
POINT	7- 8	(7600 8800)	440.00	79.0	1131.37	0.07
POINT	7- 9	(7500 8700)	449.00	88.0	1272.79	0.07
POINT	7-10	(8500 9700)	351.00	-10.0	141.42	-0.07
POINT	7-11	(8600 9800)	348.00	-13.0	282.84	-0.05
POINT	7-12	(8700 9900)	348.00	-13.0	424.26	-0.03

TABLE 5

SLOPE ON PERPENDICULAR LINE TO ARC (CONT'D.)

POINT	8- 1	(8200	9600)	361.00	7.0	141.42	0.05
POINT	8- 2	(8100	9500)	367.00	13.0	282.84	0.05
POINT	8- 3	(8000	9400)	374.00	20.0	424.26	0.05
POINT	8- 4	(7900	9300)	381.00	27.0	565.69	0.05
POINT	8- 5	(7800	9200)	388.00	34.0	707.11	0.05
POINT	8- 6	(7700	9100)	387.00	33.0	848.53	0.04
POINT	8- 7	(7600	9000)	390.00	36.0	989.95	0.04
POINT	8- 8	(7500	8900)	404.00	50.0	1131.37	0.04
POINT	8- 9	(8400	9800)	348.00	-6.0	141.42	-0.04
POINT	8-10	(8500	9900)	348.00	-6.0	282.84	-0.02
POINT	9- 1	(8100	9700)	354.00	6.0	141.42	0.04
POINT	9- 2	(8000	9600)	361.00	13.0	282.84	0.05
POINT	9- 3	(7900	9500)	367.00	19.0	424.26	0.04
POINT	9- 4	(7800	9400)	374.00	26.0	565.69	0.05
POINT	9- 5	(7700	9300)	381.00	33.0	707.11	0.05
POINT	9- 6	(7600	9200)	382.00	34.0	848.53	0.04
POINT	9- 7	(8300	9900)	348.00	0.0	141.42	0.00

The stopping rules for the arc width are a function of unit mission (unit formation) and unit type. When the unit formation is a single column, only the on-road portion of the arc is used. When the unit formation for movement is multiple column, both on-road and off-road attributes may be used. The arc width for a given perpendicular is the distance from the left side boundary to right side boundary, each established by one of the two rules.

For example, in case of a vehicle unit, an open area (code 7) and no feature data area (code 4) may be considered feasible for routes of maneuver for off-road movement, with forest, urban, marsh, water, and heath areas considered infeasible. In the case of dismounted troops, however, forest, urban, heath and open area may be considered feasible for movement routes off-road, but water and marsh areas are infeasible.

Consider the matrices shown in Tables 6 and 7. At the first cross point between the perpendicular line and y-axis grid line, it determines the code from the code of the nearest point on the y-axis grid line. If the value contained in the two dimensional array of the previous code and current code equals 'no boundary', the process continues. If the value of previous code and current code is 'boundary', it will stop. For example, a tracked vehicle can go from open area to open area but can not go from open area to water or marsh area. The dismounted unit can go from forest to forest, open or heath area, but will not go from forest to water or marsh area. That is,

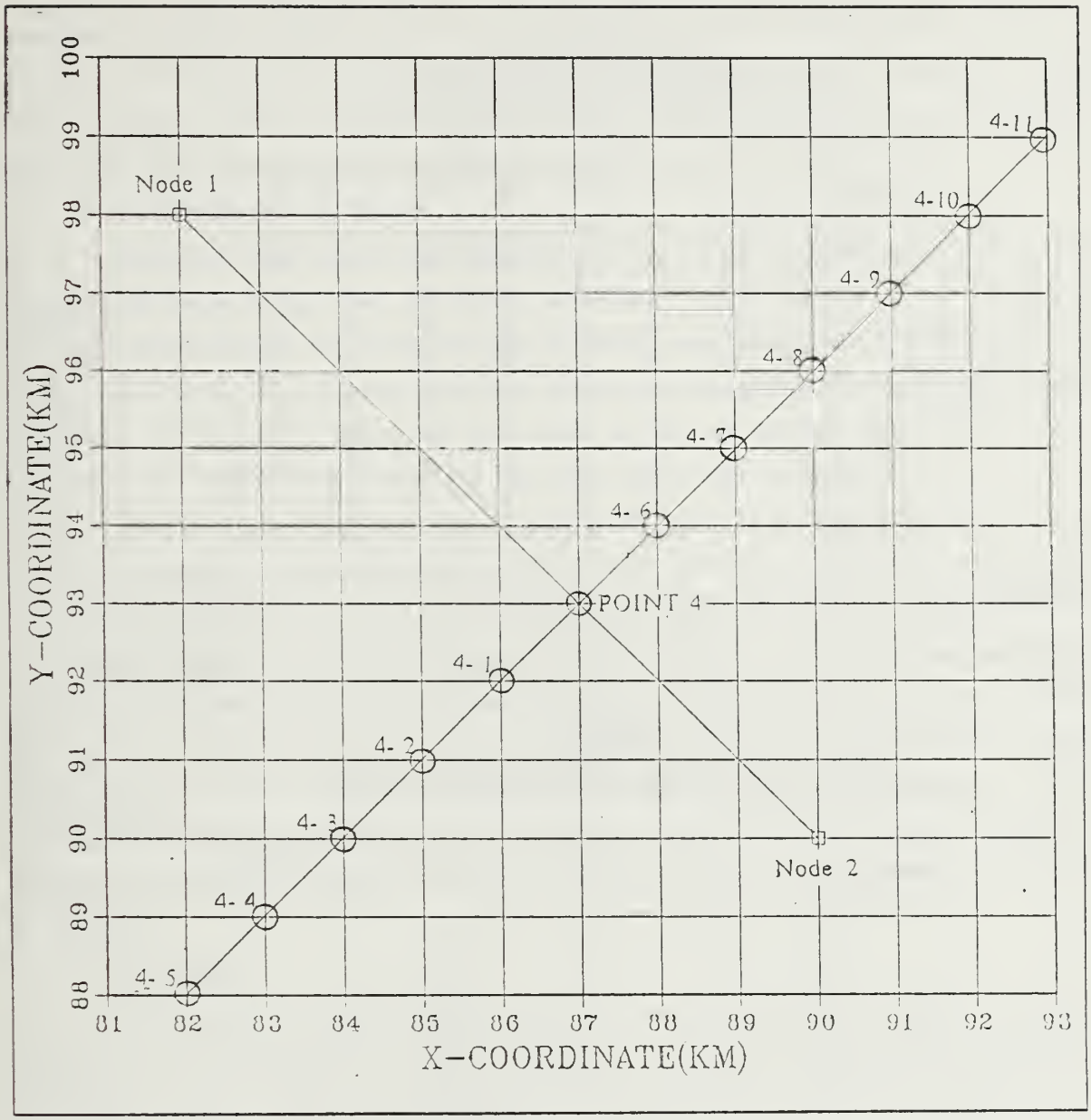

Figure 3.3 Points on the Perpendicular.

if CODE(i, j) equals 1, then the point is not a boundary, otherwise, the point is a boundary. Subscript i represents the previous code on the perpendicular line to the arc, and subscript j represents the current code on the same line as subscript i.

The slope, as well as terrain code change, could be a barrier to certain types of units. The slope of the perpendicular line to the arc affects the movement of units. The arc boundary due to slope occurs when the slope is less than a specified negative threshold or greater than a positive threshold. The threshold of slope is the

TABLE 6
BOUNDARY FOR VEHICLE UNIT

code	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	1
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	1

TABLE 7
BOUNDARY FOR DISMOUNTED TROOPS

code	0	1	2	3	4	5	6	7
0	0	1	1	0	1	0	1	1
1	0	1	1	0	1	0	1	1
2	0	1	1	0	1	0	1	1
3	0	1	1	0	1	0	1	1
4	0	1	1	0	1	0	1	1
5	0	1	1	0	1	0	1	1
6	0	1	1	0	1	0	1	1
7	0	1	1	0	1	0	1	1

maximum negotiable slope for each unit type. The threshold for vehicle units may be less than that for dismounted troops. Eventually, arc soil strength data will be used to determine these thresholds as shown in Appendix G.

Each arc width on the perpendicular line is determined by consideration of the code change rule and slope change rule simultaneously. The arc width is the distance from the stopping point on the left side to the point on the right side of the arc. The line connecting each stopping point on the left (right) side is the arc's left (right) boundary line as shown in Figure 3.4.

2. **Determine Arc Attributes**

Each arc width on the perpendicular line to the arc is determined by the code change and slope change rules previously discussed. The arc widths may be different for each perpendicular line along the arc. However, one unique arc width is required from a head node to a tail node, since it is difficult to change the unit formation during movement on an arc. Insertion of new nodes on the arc can be considered for long arcs when half of the arc is narrow but the other half of the arc is wide.

The program determines the arc width by adding the left minimum distance, the right minimum distance, and road width itself. The minimum distance on the left side is determined by the shortest perpendicular distance from the arc to the left, and similarly for the right minimum distance. When the left and right minimum distances equal zero, the arc width is considered to be the road width itself since no off-road movement is possible for that unit type. The minimum distance determination of arc width is more realistic than consideration of some average of the distances. The minimum distance has an direct effect on the flow rate of an arc. Table 8 shows an example of the minimum distance off arc, arc width and those coordinates for the plot in Figure 3.3.

3. **Flow Rate**

Using the arc width and possible speed for a specified unit type and mission, the rate of flow is computed. Possible speed is a function of arc type, unit type and unit mission type. The arc type represents several conditions of the maneuver paths between nodes and is an important element for the speed of maneuver. There are several categories of arc types as shown in Table 3 . The unit types currently considered are tracked vehicles, wheeled vehicles and dismounted troops as shown in Table 18, Appendix F. The types of unit formations (missions) currently considered are multiple column and single column as shown in Table 19, Appendix F.

The following equation is used to determine the flow rate on an arc and is measured in terms of battalions per hour for each unit type [Ref. 5: p. 20].

$$FR_j = (SP_{bn} \times W_j) / (DW_{bn} \times DD_{bn}) \qquad \text{(eqn 3.1)}$$

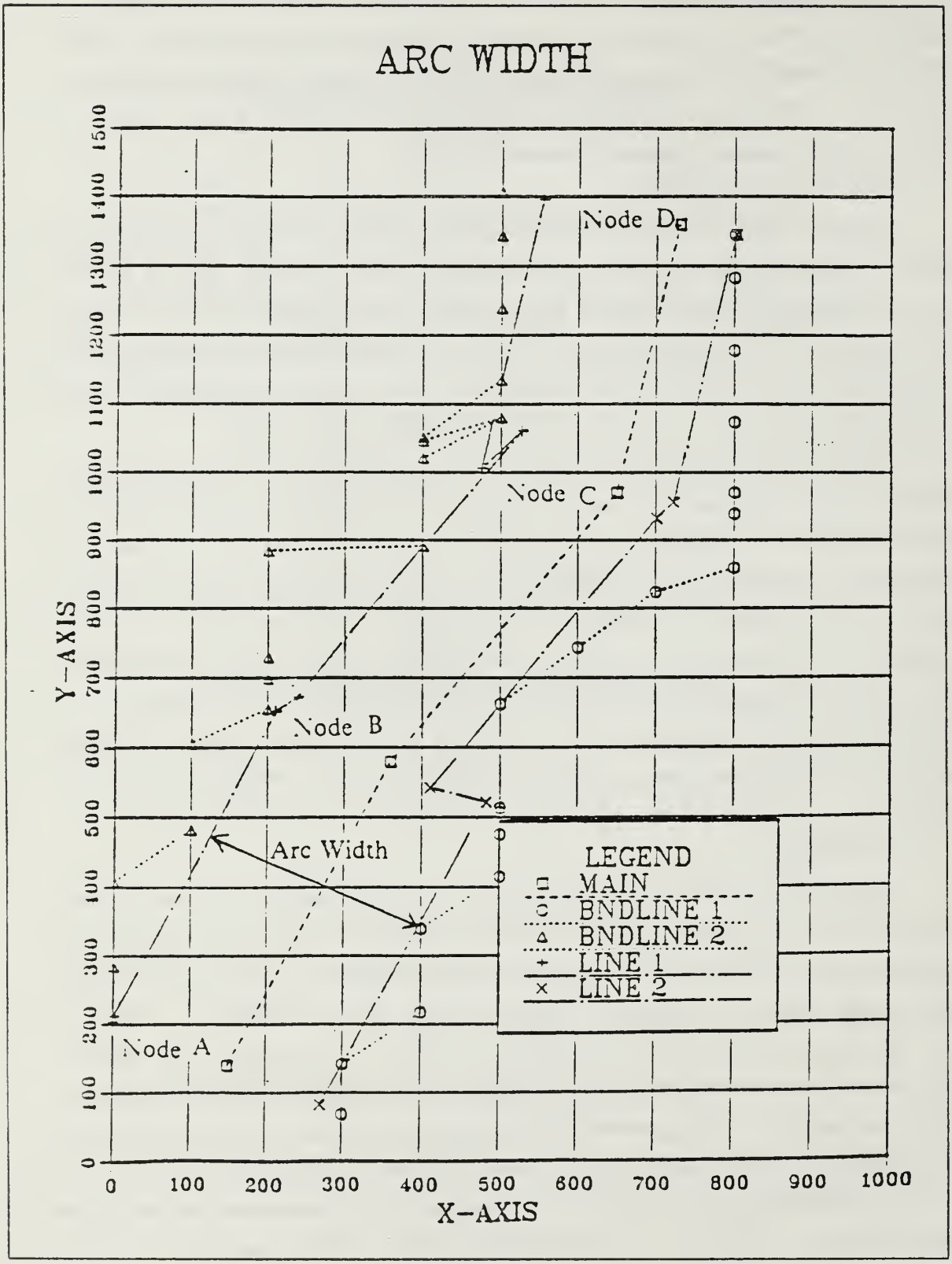

Figure 3.4 Arc Width.

TABLE 8

MINIMUM DISTANCE OFF ARC

		MINIMUM DISTANCE OFF ARC			
LEFT MIN DISTANCE	:	141.4			
RIGHT MIN DISTANCE	:	141.4			
ROAD WIDTH	:	6.0			
ARC WIDTH	:	288.8			
		POINT 1		POINT 2	
		X LEFT	Y LEFT	X RIGHT	Y RIGHT
INITIAL POINT	:	8900.0	8900.0	9099.0	9100.0
END POINT	:	8100.0	9700.0	8299.0	9900.0

The equation is used for all arcs, j, to compute the rate of flow, FRj, based on the speed of a battalion sized unit, SP_{bn}, the width of the arc, W_j, the doctrinal width of the battalion, DW_{bn}, and its doctrinal depth, DD_{bn}. The doctrinal width and depth of the battalion depend on the unit type. Since the doctrinal width and depth of a battalion of dismounted troops is different from those of a tank battalion, different values were assigned for these parameters as a function of unit type. Table 9 shows an example of flow rate and maximum number of elements on the arc width for Arc 1 (from Node 1 to Node 2) as shown in Figure 3.2.

TABLE 9

FLOW RATE

ARC TYPE	:	2(Concrete & Asphalt Road)	
UNIT TYPE	MISSION TYPE	FLOW RATE	MAX # ELEMENT
1	1	2.41	11
1	2	10.00	1
2	1	3.37	14
2	2	11.25	1

Unit Type 1 : Tracked Vehicle Unit
Unit Type 2 : Wheeled Vehicle Unit
Mission Type 1 : Multiple Column Formation
Mission Type 2 : Single Column Formation

The maximum number of elements which can move in multiple column formation on the arc width is computed. The number is determined from the arc width and the distance between elements (i.e., the maximum number of elements equal the arc width divide by the specified distance between elements for each formation type).

4. Arc Traversal Time

For purposes of this planning model, it is not necessary to develop extremely accurate representations of attacking force movement. General estimates of expected movement rates based on unit type and the characteristics of the terrain being crossed are sufficient. For simplicity, arc traversal time is the arc length divided by arc speed for the attacking force unit.

Assume the formation width of a single column is less than the width of road itself. If the width of the unit formation is less than the width of the road itself, then arc speed is on-road speed. Otherwise, arc speed is off-road speed. On-road speed is assumed to be larger than off-road speed. Arc speeds, in terms of on-road and off-road speeds, are provided in Appendix A.

IV. NETWORK IMPLEMENTATION

A. INTRODUCTION

Methodology for determination of minimum time and minimum distance paths using a network representation of terrain is presented in this chapter. This chapter is related to the computer program for the Cartesian space network in Appendix D. The Cartesian space network is used to find a shortest path between two given nodes on an undirected graph represented by a list of arcs. Input data for this network program is the output of the computer program of single arc attributes given in Appendix C.

B. DATA FILE

The network representation of the maneuver area is placed on two disk files which are the output files from the computer program for single arc attributes: one for vehicle units and one for dismounted troops. Eight characteristics are used to represent an arc: arc identification, tail node identification, head node identification, traversal time, flow rate, length, width, and arc speed. These characteristics provide sufficient data for the algorithms to determine the minimum time or distance path from starting node to terminal node. The time for the arc is an integer representing the value of traversal time for the movement of each unit. The flow rate is an integer value representing flow rate for arc, j, times one hundred ($FR_j \times 100$). The integer value is used for computational reasons in the shortest path algorithm. The length is the distance between head node and tail node in kilometers. The width of the arc is the width of road itself plus the left and right off-road widths in kilometers for which the terrain will support movement. The arc speed represents the average speed for each unit along the arc.

Two kinds of width determination rules are used for the input data files from the single arc attributes program. The width determination rule is used in the following situation. Recall from Chapter III that either a terrain code or slope change may terminate the search for the arc width on each side of the arc. If the first point on a perpendicular to an arc terminates the search, the width determination rule is invoked. For example, suppose the distance to the first point is 100 meters. If rule 1 is used, the off-road width is assigned to be 100 meters for that perpendicular. If it is desired to consider off-road width to be zero, then Rule 2 is used.

For example, if the unit was a convoy of wheeled vehicles, then Rule 2 would likely be selected. For tracked vehicles or dismounted troops, it may be appropriate to select Rule 1. Table 10 shows an example of the data for the network representation of vehicle unit from width determination rule 1.

TABLE 10

DATA FOR VEHICLE UNIT - RULE 1

ARC	TAIL	HEAD	TIME Unit*	FLOW Unit**	DIST Km	WIDTH Km	SPEED Km/Hr
1	1	2	5	240	1.1	0.289	25
2	1	8	6	94	1.4	0.113	25
3	2	3	8	106	2.0	0.128	25
4	2	9	58	1	2.6	0.010	5
5	2	10	76	20	3.5	0.120	5
6	3	4	9	97	2.2	0.117	25
7	3	10	56	20	2.5	0.123	5
8	4	5	5	144	1.3	0.174	25
9	4	10	8	102	1.9	0.123	25
10	4	11	9	76	1.8	0.115	20
11	5	6	4	184	1.1	0.222	25
12	5	11	7	89	1.3	0.134	20
13	5	12	5	102	1.3	0.123	25
14	6	12	8	98	1.9	0.118	25
15	6	7	8	103	1.6	0.155	20
16	7	12	12	84	2.2	0.127	20
17	7	13	10	70	1.8	0.106	20
18	8	9	9	71	1.7	0.107	20
19	8	18	40	17	1.8	0.108	5
20	9	10	59	28	2.7	0.170	5
21	9	14	35	19	1.6	0.115	5
22	9	18	48	24	2.2	0.148	5
23	9	19	52	19	2.4	0.115	5
24	10	11	5	171	1.3	0.206	25
25	10	14	8	80	1.6	0.121	20
26	10	15	37	20	1.7	0.123	5
27	11	12	6	70	1.2	0.105	20
28	11	15	7	93	1.6	0.112	25
29	11	16	8	115	1.9	0.138	25
30	12	13	8	96	1.6	0.145	20
31	12	16	6	94	1.5	0.114	25
32	13	16	43	20	2.0	0.125	5
33	13	17	40	20	1.8	0.124	5
34	14	15	7	119	1.6	0.144	25
35	14	19	47	23	2.1	0.140	5

*Unit** : *Hour* × *100*

*Unit*** : (*Battalions* / *Hour*) × *100*

TABLE 10

DATA FOR VEHICLE UNIT - RULE 1 (CONT'D.)

36	15	16	8	70	1.5	0.105	20
37	15	21	5	76	1.0	0.115	20
38	16	17	59	17	2.7	0.105	5
39	16	23	9	171	2.1	0.206	25
40	16	24	42	21	1.9	0.130	5
41	17	24	5	108	1.3	0.130	25
42	18	19	57	18	2.6	0.110	5
43	18	26	58	24	2.6	0.147	5
44	18	31	95	17	4.3	0.107	5
45	19	20	5	240	1.1	0.289	25
46	19	26	27	48	1.3	0.292	5
47	20	21	9	119	2.2	0.143	25
48	20	22	8	136	2.0	0.164	25
49	20	27	7	105	1.7	0.126	25
50	20	28	45	24	2.1	0.147	5
51	21	22	21	20	1.0	0.124	5
52	22	23	6	89	1.2	0.134	20
53	22	29	7	101	1.3	0.153	20
54	23	24	7	121	1.7	0.146	25
55	23	25	8	111	1.8	0.133	25
56	23	29	3	121	0.8	0.146	25
57	24	25	8	78	1.6	0.118	20
58	25	30	6	114	1.1	0.172	20
59	26	27	33	21	1.5	0.129	5
60	26	31	65	21	3.0	0.131	5
61	27	28	42	33	1.9	0.202	5
62	27	31	14	115	2.7	0.173	20
63	27	32	7	93	1.7	0.113	25
64	27	33	58	24	2.6	0.147	5
65	28	29	43	47	2.0	0.285	5
66	28	33	39	33	1.8	0.202	5
67	28	34	47	18	2.2	0.110	5
68	29	34	10	91	2.3	0.110	25
69	30	34	9	113	2.2	0.137	25
70	31	32	11	99	2.6	0.120	25
71	32	33	9	85	1.7	0.128	20
72	33	34	49	28	2.2	0.170	5

C. DETERMINING THE PARAMETERS

The program requires several parameters for determining the shortest distance or time path. Changeable parameters are unit type, unit formation, starting point and terminal point, arc travel time with obstacles, and whether minimum time or distance is to be calculated. During the program runs, the parameters are provided by user interaction with the program.

1. **Unit and Path**

The path can be selected in two ways: minimum time or minimum distance. Minimum time determines the least time route through a network from a user selected starting node, ISTART, to a terminal node, LAST. Minimum distance determines the shortest distance path between two selected nodes.

The unit type is vehicle unit or dismounted troops based on a battalion sized unit. Vehicle unit is assumed to be tracked vehicles for the examples described later. When the unit is determined, the program reads the data file appropriate for the selected unit.

2. **Unit Formation**

The unit formation is described as either a single or multiple column formation. For dismounted troops, a single column formation is defined as two columns, with multiple column formation being more than two columns.

The width of the unit formation is the number of columns times the input spacing between columns. The depth of the unit formation is the number of rows times the input spacing between rows. If the width of the formation is larger than the arc width, then the program assigns a "large" number to the travel time, which is the arc cost in the shortest path algorithm. Otherwise, the program uses the original value for time previously determined as the arc cost.

3. **Obstacles**

Obstacles (e.g., minefields) can change the travel time and the shortest path in the program. The program requires information about obstacles on the arcs. If there is an obstacle on an arc, then the travel time is appropriately increased. For the purpose of examples in this thesis, the original travel time is multiplied by four when obstacles are present.

D. SHORTEST PATH ALGORITHM

The determination of the minimum time and distance path through the sub-network is one of a class of many such problems often referred to as "shortest path" network problem. The general shortest path problem is to determine the least cost route through a network starting at node ISTART and ending at node LAST. The cost of a route is some function of the characteristics of the arcs and nodes that make up the route from node ISTART to node LAST. In the case of this minimum time path problem, the cost of traversing an arc is the amount of time it takes a unit to

traverse the arc. The cost of the route is the sum of the costs of all the arcs in the path from the supply node to the terminal node.

Before selecting a method of finding the shortest path, the network must be examined to insure that there is, in fact, a solution in all cases. Because no arc in the network will have a negative cost associated with it, the addition of another arc to any path will never reduce the total time traveled. Therefore, negative cycles cannot exist in the network. The only arcs which have a zero cost are those leaving the supply node and entering the terminal node. Furthermore, boundaries are selected to run parallel to avenues of approach through a sector, so it is valid to assume that there will be at least one set of connected nodes which extends laterally through the sector. Thus, the method of construction of the sub-networks insures there will always exist at least one path between supply and terminal nodes. Therefore, there must be a solution to the shortest path problem, and there will not be any cycles in a minimum path(no node will be visited twice). If the assumption that boundaries are drawn which include a connected path from the start to the terminal node is invalid, a solution may not exist to the problem.

Because of the many applications for this class of problems, there are many algorithms available for its solution. Most of these algorithms consist of two procedures: a label correcting procedure and a search procedure.

In the label correcting procedure each node is initially assigned an infinite cost with the exception of the starting node, s which is given a cost of zero. Letting c(u) be the cost assigned to node u, d(u,v) be the cost to traverse the arc from node u to node v, and pred(u) be the node previous to u in a path, then the following rule is applied to change the costs associated with node v:

$$\text{If} \quad c(u) + d(u,v) < c(v) \qquad \text{(eqn 4.1)}$$
$$\text{then} \quad c(v) = c(u) + d(u,v) \text{ and } pred(v) = u.$$

Though inefficient if applied indiscriminately, if this rule is continuously applied until no cases of Equation 4.1 being true can be found, the chain of pred(v) to pred(u) = s will describe the minimum path from every node in the network to the starting node, s. [Ref. 6: p. 61]

The search procedure is used to decide in which order the nodes are to be scanned to apply the labeling rule. This step is where most algorithms differ and also

where the efficiency of the algorithm is achieved. In most cases one of three search techniques are used: Dijkstra's algorithm, depth first search, and breadth first search, but many hybrid techniques are also available. Dijkstra's algorithm gives priority of search to the adjacent node which is the shortest distance away. Depth first search gives priority to the most recent node searched. Breadth first search gives priority to the oldest node searched. The Dijkstra's algorithm was selected for use in the program because all arc lengths are at least greater than zero and are not the same.

As the name implies, the shortest path algorithm solves arc cost in terms of the minimum travel time and distance for an attacker across an arc. For simplicity, the attacker force is treated as a rectangle for all computations and travel time is the arc length divided by average arc speed of the attacking force. The former is an arc attribute, while the latter is determined from unit movement capabilities. The algorithm is described below.

Input: Cartesian Space network G(V,E), nodes V, arcs E with their characteristics as described in Section B.

output: Shortest path distances or times(d(v)) from starting node, s, to terminal node, t.

Step 1. Initialization:

a) set all labels to a very large value.

$d(v) = \text{infinity}$

b) set cost of starting node to zero.

$d(s) = 0$

c) place starting node in the Set S.

$\text{Set } S = \{s\}$

b) remove the Set S from Set V, the set of nodes of G.

$\text{Set } T = \text{Set } V - \{s\}$

Step 2. For each node v adjacent to the node u (s.t. $u \in S$, $v \in T$).

{ If $d(u) + l(u,v) < d(v)$

then $d(v) = d(u) + l(u,v)$ }

Step 3. Determination of Set S and Set T.

a) determine V_{min}

$V_{min} = \text{argmin } \{d(v)\}$

b) save the value of cost {d(v)} in the Set S.

$\text{Set } S = \text{Set } S + \{V_{min}\}$

c) remove the Set S from the set of nodes of G (Set T).

Set T = Set T - {s}

d) If all arcs in the graph have not been analyzed, go to Step 2.

Step 5. Return Set S to the minimum time path solution routine. Stop.

l(u,v) is the length of the arc between node u and adjacent node v.

Set S is the set of nodes such that d(v) gives the true shortest path length from starting node s to some node v.

Set T is Set V - Set S

Set V is the set of nodes of undirected graph G. [Ref. 7: p. 129]

When Set T is empty, each node in the network will have a cost assigned which represents the minimum cost for all possible paths from the node to the starting node,s. The minimum path from any node to the start node can then be found by tracing the chain of predecessor values back to the start node. When the algorithm is completed, the minimum time or distance path is defined by the chain of predecessor values from the supply node,s, to the terminal node. After determining the node number along the minimum path, the arc number is found for computing total travel time or total distance from the start node to the terminal node. The total travel time for the unit is computed as follows.

$$T_{unit} = (\sum T_{arc\ i}) + (UL / SP) \qquad \text{(eqn 4.2)}$$

The equation is used for total travel time for a unit, T_{unit}, based on the total travel time for an element(summation of $T_{arc\ i}$) and unit length, UL, divided by average speed along the path, SP. The second term of Equation 4.2 represents additional travel time for the entire unit to complete the path.

The following equation is used to determine average speed along the path.

$$SP = \sum (d_i \times V_i) / D \qquad \text{(eqn 4.3)}$$

The average speed SP is computed based on the summation of length of arc_i, d_i, multiplied by the speed on arc_i, V_i, and that quantity divided by the total path length, D. This algorithm finds the minimum time or distance path through the sector by using the unit formation. Any type of arc can be on this path and speeds used to determine travel time are adjusted for the type of arcs along the minimum time path. Table 11 shows an example of results for the vehicle unit from the network program given in Appendix D.

TABLE 11

OUTPUT FOR VEHICLE UNIT - RULE 1

```
-----------------------------------------------
MIN  TIME  PATH
-----------------------------------------------
VEHICLE UNIT
FORMATION : 50 ROWS  1 COLUMNS
                ROW SPACE 50 METERS
NO MINEFIELD ARC
-----------------------------------------------
SUM OF TRAVERSAL TIME :   0  HOUR   36  MINUTE
TOTAL TRAVERSAL TIME  :   0  HOUR   42  MINUTE
NODE NUMBER ALONG MIN TIME PATH
         N    NODE
         1     1
         2     2
         3     3
         4     4
         5     5
         6    12
         7    16
         8    23
         9    29
        10    34
ARC NUMBER ALONG MIN TIME PATH
         N    ARC
         1     1
         2     3
         3     6
         4     8
         5    13
         6    31
         7    39
         8    56
         9    68
TOTAL PATH  LENGTH    :   14.60  KM
-----------------------------------------------
```

V. ANALYSIS

A. INTRODUCTION

This chapter describes the results of varying parameter values of the single arc attributes program and Cartesian Space Network program used in determining minimum distance or minimum time path. The results in this chapter are final outputs from the single arc attributes program through the Cartesian Space Network program. A 10 X 10 Km sector of European terrain is used for the analysis to demonstrate the algorithms. There are 34 nodes and 72 arcs in the transportation network. The start node is Node 1 and the terminal node is Node 34. The unit size is battalion level consisting of 50 tanks in a tracked vehicle unit and 1000 men in a dismounted troop unit. Minimum distance path is first discussed, followed by a comparative analysis of the minimum time path generated under various conditions.

B. MINIMUM DISTANCE PATH

This algorithm determines the path that provides the shortest distance through the network. The arc cost is defined by the distance between head node and tail node. The tactical significance of this algorithm would be the determination of the path through the network that provides the best route to reach the given point in terms of distance. A path of this nature might be used by raiding or reconnaissance elements of a dismounted troop unit.

The minimum distance path is Path D through the network shown in Figure 5.1 The distance of the path is 10.9 Km from the start node (Node 1) to the terminal node (Node 34). Figure 5.1 shows the minimum distance path and several minimum time paths described in the next section.

C. MINIMUM TIME PATH

The purpose of this section is to demonstrate the full capabilities of the algorithms developed in terms of minimum time paths. The following parameters are considered in the analysis:

- Unit type - vehicles and dismounted troops.
- Formation size - specified by number of rows and columns in the formation.
- Width determination rule - specifies the method used to determine total arc width in the special case described in Chapter IV.
- Obstacle (minefield) representation.

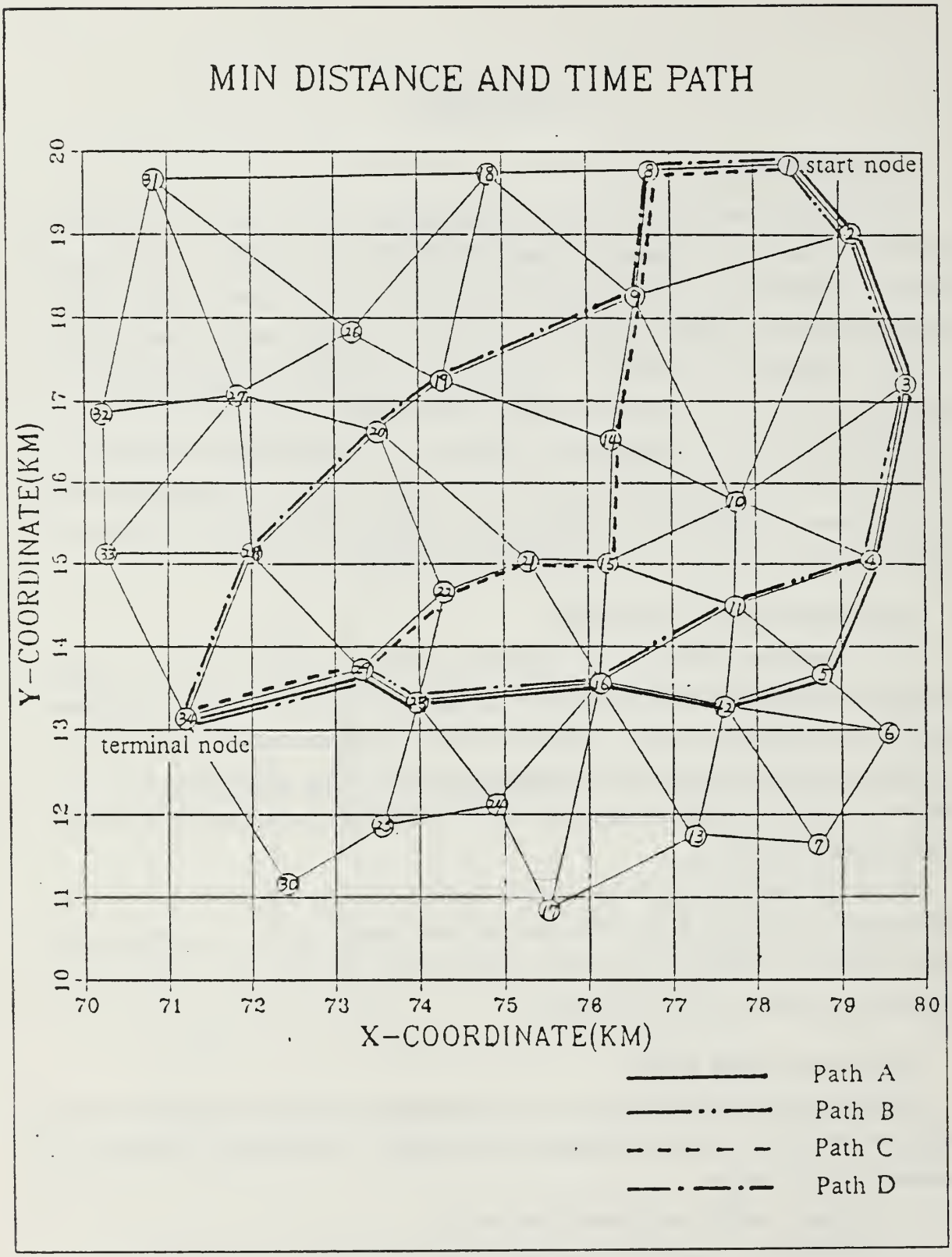

Figure 5.1 Min Distance and Time Path.

- Bridge representation.

The results of several model runs for various parameter values are given in Tables 12 and 13. The total travel time for the lead element and the unit, the path length, and the minimum time path from Figure 5.1 are presented for each case. The impact of the width determination rule selected is shown in Table 12 for vehicle units and in Table 13 for dismounted troop units.

TABLE 12

MIN TIME PATH - VEHICLE UNIT

Case	Formation Row x Col	Rule 1,2	Mine y/n	Bridge y/n	Travel Time element/unit	Length Km	Path
A.1	50 X 1	1	no	no	36 / 42	14.6	A
A.2	25 X 2	1	no	no	36 / 39	14.6	A
A.3	10 X 5	1	no	no	36 / 36	14.6	A
A.4	5 X 10	1	no	no	no feasible	route	
B.1	50 X 1	1	yes	no	36 / 43	14.2	B
B.2	25 X 2	1	yes	no	36 / 39	14.2	B
B.3	10 X 5	1	yes	no	36 / 36	14.2	B
B.4	5 X 10	1	yes	no	no feasible	route	
C.1	50 X 1	1	no	yes	36 / 43	14.2	B
C.2	25 X 2	1	no	yes	36 / 40	14.2	B
C.3	10 X 5	1	no	yes	36 / 36	14.2	B
C.4	5 X 10	1	no	yes	no feasible	route	
D.1	50 X 1	2	no	no	36 / 42	14.6	A
D.2	25 X 2	2	no	no	no feasible	route	
D.3	10 X 5	2	no	no	no feasible	route	
D.4	5 X 10	2	no	no	no feasible	route	
E.1	50 X 1	2	yes	no	36 / 43	14.2	B
E.2	25 X 2	2	yes	no	no feasible	route	
E.3	10 X 5	2	yes	no	no feasible	route	
E.4	5 X 10	2	yes	no	no feasible	route	
F.1	50 X 1	2	no	yes	36 / 43	14.2	B
F.2	25 X 2	2	no	yes	no feasible	route	
F.3	10 X 5	2	no	yes	no feasible	route	
F.4	5 X 10	2	no	yes	no feasible	route	

1. **Vehicle Unit**

The formation of vehicle units is either a single column or multiple column formation. The width of the formation for a single column is five meters, while that for the multiple column formation is 25 meters for each column. For example, the width of a 50 X 1 formation is five meters, of a 25 X 2 formation is 25 meters, and of a 10 X 5 formation is 100 meters. Note from Table 12 that for vehicle units, Path A and Path B are selected for the minimum time path under various situations, because these paths do not have arcs which are in a forest or marsh area.

a. No Minefield

For the no minefield and no bridge situation, the minimum time path is Path A in the two cases which use width determination rules 1 and 2 (see Cases A.1, A.2, A.3, and D.1). The path from the results of Rule 1 yields an optimal route for movement until the formation size is 10 X 5 (i.e., formation width is 100 meters in which case no feasible route exists(see Case A.4 in Table 12)).

For Rule 2, a path exists only for the single column formation (Case D.1). For wider formations, there is no feasible route for movement (see Cases D.2, D.3, D.4 in Table 12). The distance of the minimum time path is 14.6 Km and travel time for an element is 36 minutes, as shown in Table 12 . Note that the unit travel times vary as a function of formation size.

b. Minefield

The purpose of runs labeled Cases B and E in Table 12 is to demonstrate the effect of insertion of obstacles on the network. For these runs, a minefield is positioned on Arc 8 which is between node 5 and node 12 in Figure 5.1. Path B bypasses the arc which has a minefield for both Rule 1 and 2. The reason is that, for these runs, arc cost is four times larger when a minefield is on the arc as compared to the no minefield case (i.e., the travel time is four times larger for a minefield arc).

Other than a change from Path A to Path B, the effects of the width determination rule and formation size are similar to the no minefield case.

c. Bridges

Bridges tend to constrain movement of units to only the on-road portion of the arc. For these runs (shown as Cases C and F in Table 12), a bridge was inserted on arc 8 instead of a minefield. Even though Rule 1 (Case C runs) are included, only Rule 2 would be applied in the real situation. The minimum time path bypasses the arc which has the bridge for both rules. The width of the bridge is assumed to be four

meters. Because the width of the bridge is less than the width of the formation in all cases, the minimum time path algorithm does not include the bridge arc on the optimal path. The results for this case are the same as for the minefield case.

2. **Dismounted Troop Units**

The formations for dismounted troop units are considered as either one double column (six meters wide) or multiple column (five meters for each column). For example, a 500 X 2 formation is six meters wide, while a 50 X 20 formation is 100 meters wide. For a comparison of vehicle versus troop units with the same formation widths, consider Cases A.2, G.2 and D.2, F.2 in Tables 12 and 13. Note that under Rule 1, Path C is optimal for troop units (Case G.2) while Path B is optimal for vehicle units (Case A.2). For the wider formations (Cases G.4 through G.6) the optimal path shifts to Path B under Rule 1. For the tighter arc width restrictions of Rule 2, Path B is optimal in all cases for dismounted troop units.

Finally, the effect of long units for travel time is evident from Case G.1. Note that an additional 44 minutes is required for the entire unit to complete Path C.

This chapter has demonstrated various aspects of the algorithms developed in this thesis. Except for those areas described in Chapter VI, the model is ready for use in a full production mode for any terrain areas which have digitized terrain data and transportation networks in the form described in Chapter III.

TABLE 13

MIN TIME PATH - DISMOUNTED TROOP UNITS

Case	Formation Row x Col	Rule 1,2	Mine y/n	Bridge y/n	Travel Time element/unit	Length Km	Path
G.1	500 X 2	1	no	no	4:22 / 5:06	11.9	C
G.2	200 X 5	1	no	no	4:22 / 4:39	11.9	C
G.3	50 X 20	1	no	no	4:22 / 4:27	11.9	C
G.4	20 X 50	1	no	no	4:40 / 4:43	14.2	B
G.5	10 X 100	1	no	no	4:40 / 4:42	14.2	B
G.6	7 X 140	1	no	no	4:40 / 4:41	14.2	B
F.1	500 X 2	2	no	no	4:40 / 5:31	14.2	B
F.2	200 X 5	2	no	no	4:40 / 5:00	14.2	B
F.3	50 X 20	2	no	no	4:40 / 4:45	14.2	B
F.4	20 X 50	2	no	no	4:40 / 4:43	14.2	B
F.5	10 X 100	2	no	no	4:40 / 4:42	14.2	B
F.6	7 X 140	2	no	no	4:40 / 4:41	14.2	B

VI. SUMMARY AND FUTURE RESEARCH AREAS

It is essential that any combat simulations of the Airland battle provide a realistic representation of terrain and environment. Previous approaches have typically utilized either hex or grid squares within the model to describe terrain conditions. In some cases network overlays were used for convoy movements, but no models exist which capture essential terrain characteristics without the internal use of some grid pattern.

In order to take advantage of the various algorithms available using networks without the need for gridded terrain representation, it is necessary to describe essential terrain parameters as arc and node attributes. A family of algorithms were developed and demonstrated in this thesis which accomplishes this objective. These algorithms can be implemented for any terrain area for which appropriate data exists. The resulting network is then usable in a model such as ALARM for both the planning and execution phases of combat representation.

Areas of future research include the development of a maximum flow path algorithm. This area requires a significant effort in determining appropriate arc costs and development of maximum flow methodology for an undirected network. In addition, several refinements are possible to represent other aspects of terrain characteristics not explicitly contained in the current model. Finally, additional test runs of the model with different terrain areas, unit types, combat formations,and boundary rules need to be made.

APPENDIX A
ARC WIDTH AND SPEED

The detailed model output for the network shown in Figure 5.1 is given in this appendix.

TABLE 14
VEHICLE UNIT - RULE 1

Arc	Node H	Node T	Road Width	Off-Road Left	Off-Road Right	Total Arc Width	Speed On	Speed Off
1	1	2	6	141	141	288	40	25
2	1	8	6	100	7	113	40	25
3	2	3	6	109	12	127	40	25
4	2	9	2	4	4	10	5	5
5	2	10	2	108	10	120	5	5
6	3	4	6	111	0	117	40	25
7	3	10	2	1	120	123	5	5
8	4	5	6	112	56	174	40	25
9	4	10	6	111	7	123	40	25
10	4	11	5	6	104	115	30	20
11	5	6	6	133	83	222	40	25
12	5	11	4	119	11	134	30	20
13	5	12	6	9	108	123	40	25
14	6	12	6	101	11	118	40	25
15	7	6	4	129	22	155	30	20
16	7	12	4	123	0	127	30	20
17	7	13	4	101	0	106	30	20
18	8	9	5	102	0	107	30	20
19	8	18	2	6	100	108	5	5
20	9	10	2	112	56	170	5	5
21	9	14	2	101	13	115	5	5
22	9	18	2	137	9	148	5	5
23	9	19	2	5	108	115	5	5
24	10	11	6	100	100	206	40	25
25	10	14	5	115	1	121	30	20
26	10	15	2	8	113	123	5	5
27	11	12	5	100	0	105	30	20
28	11	15	6	105	0	112	40	25
29	11	16	6	15	117	138	40	25
30	12	13	4	141	0	145	30	20

Unit of on(off)-road speed : *Km / Hr*

Units of several widths : *Meters*

TABLE 14

VEHICLE UNIT - RULE 1 (CONT'D.)

31	12	16	6	101	7	114	40	25
32	13	16	2	116	7	125	5	5
33	13	17	2	7	114	124	5	5
34	14	15	6	100	38	144	40	25
35	14	19	2	122	16	140	5	5
36	15	16	5	100	0	105	30	20
37	15	21	4	100	10	115	30	20
38	16	17	2	103	0	105	5	5
39	16	23	6	100	100	206	40	25
40	16	24	2	128	0	130	5	5
41	17	24	6	114	10	130	40	25
42	18	19	2	108	0	110	5	5
43	18	26	2	138	7	147	5	5
44	18	31	2	5	100	107	5	5
45	19	20	6	141	141	289	40	25
46	19	26	2	228	62	292	5	5
47	20	21	6	130	8	143	40	25
48	20	22	6	109	49	164	40	25
49	20	27	6	107	13	126	40	25
50	20	28	2	16	128	147	5	5
51	21	22	2	12	109	124	5	5
52	22	23	5	103	26	134	30	20
53	22	29	5	14	134	153	30	20
54	23	24	6	117	23	146	40	25
55	23	25	6	115	13	133	40	25
56	23	29	6	109	31	146	40	25
57	24	25	5	13	101	118	30	20
58	25	30	4	56	112	172	30	20
59	26	27	2	9	117	129	5	5
60	26	31	2	119	10	131	5	5
61	27	28	2	100	100	202	5	5
62	27	31	5	112	56	173	30	20
63	27	32	6	6	101	113	40	25
64	27	33	2	138	7	147	5	5
65	28	29	2	141	141	285	5	5
66	28	33	2	100	100	202	5	5
67	28	34	2	108	0	110	5	5
68	29	34	6	0	104	110	40	25
69	30	34	6	118	12	137	40	25
70	31	32	6	114	0	120	40	25
71	32	33	4	100	24	128	30	20
72	33	34	2	112	56	170	5	5

TABLE 15

VEHICLE UNIT - RULE 2

Arc	Node H	Node T	Road Width	Off-Road Left	Off-Road Right	Total Arc Width	Speed On	Speed Off
1	1	2	6	0	0	6	40	25
2	1	8	6	0	0	6	40	25
3	2	3	6	0	0	6	40	25
4	2	9	2	0	0	2	5	5
5	2	10	2	0	0	2	5	5
6	3	4	6	0	0	6	40	25
7	3	10	2	0	0	2	5	5
8	4	5	6	0	0	6	40	25
9	4	10	6	0	0	6	40	25
10	4	11	5	0	0	5	30	20
11	5	6	6	0	0	6	40	25
12	5	11	4	0	0	4	30	20
13	5	12	6	0	0	6	40	25
14	6	12	6	0	0	6	40	25
15	7	6	4	0	0	4	30	20
16	7	12	4	0	0	4	30	20
17	7	13	4	0	0	4	30	20
18	8	9	5	0	0	5	30	20
19	8	18	2	0	0	2	5	5
20	9	10	2	0	0	2	5	5
21	9	14	2	0	0	2	5	5
22	9	18	2	0	0	2	5	5
23	9	19	2	0	0	2	5	5
24	10	11	6	0	0	6	40	25
25	10	14	5	0	0	5	30	20
26	10	15	2	0	0	2	5	5
27	11	12	5	0	0	5	30	20
28	11	15	6	0	0	6	40	25
29	11	16	6	0	0	6	40	25
30	12	13	4	0	0	4	30	20
31	12	16	6	0	0	6	40	25
32	13	16	2	0	0	2	5	5
33	13	17	2	0	0	2	5	5
34	14	15	6	0	0	6	40	25
35	14	19	2	0	0	2	5	5
36	15	16	5	0	0	5	30	20
37	15	21	4	0	0	4	30	20
38	16	17	2	0	0	2	5	5
39	16	23	6	0	0	6	40	25
40	16	24	2	0	0	2	5	5
41	17	24	6	0	0	6	40	25
42	18	19	2	0	0	2	5	5
43	18	26	2	0	0	2	5	5
44	18	31	2	0	0	2	5	5
45	19	20	6	0	0	6	40	25
46	19	26	2	0	0	2	5	5
47	20	21	6	0	0	6	40	25
48	20	22	6	0	0	6	40	25
49	20	27	6	0	0	6	40	25
50	20	28	2	0	0	2	5	5

TABLE 15

VEHICLE UNIT - RULE 2 (CONT'D.)

51	21	22	2	0	0	2	5	5
52	22	23	5	0	0	5	30	20
53	22	29	5	0	0	5	30	20
54	23	24	6	0	0	6	40	25
55	23	25	6	0	0	6	40	25
56	23	29	6	0	0	6	40	25
57	24	25	5	0	0	5	30	20
58	25	30	4	0	0	4	30	20
59	26	27	2	0	0	2	5	5
60	26	31	2	0	0	2	5	5
61	27	28	2	0	0	2	5	5
62	27	31	5	0	0	5	30	20
63	27	32	6	0	0	6	40	25
64	27	33	2	0	0	2	5	5
65	28	29	2	0	0	2	5	5
66	28	33	2	0	0	2	5	5
67	28	34	2	0	0	2	5	5
68	29	34	6	0	0	6	40	25
69	30	34	6	0	0	6	40	25
70	31	32	6	0	0	6	40	25
71	32	33	4	0	0	4	30	20
72	33	34	2	0	0	2	5	5

TABLE 16

DISMOUNTED TROOP UNIT - RULE 1

Arc	Node H	Node T	Road Width	Off-Road Left	Off-Road Right	Total Arc Width	Speed On	Speed Off
1	1	2	6	990	141	1137	4	3
2	1	8	6	100	7	113	4	3
3	2	3	6	1204	218	1428	4	3
4	2	9	2	4	105	111	4	3
5	2	10	2	1187	975	2164	4	2
6	3	4	6	1118	203	1327	4	2
7	3	10	2	1202	160	1364	4	3
8	4	5	6	1230	670	1906	4	2
9	4	10	6	1216	59	1280	4	3
10	4	11	5	105	1042	1152	4	3
11	5	6	6	1462	664	2132	4	3
12	5	11	4	1304	1078	2386	4	3
13	5	12	6	1092	1083	2181	4	3
14	6	12	6	1114	916	2036	4	3
15	7	6	4	1185	538	1727	4	3
16	7	12	4	1357	1110	2471	4	3
17	7	13	4	1115	912	2032	4	3
18	8	9	5	113	0	118	4	3
19	8	18	2	6	77	85	4	2
20	9	10	2	1230	1062	2294	4	2
21	9	14	2	1109	920	2030	4	2
22	9	18	2	137	34	173	4	2
23	9	19	2	5	108	115	4	2
24	10	11	6	1100	1000	2106	4	3
25	10	14	5	1267	1037	2309	4	3
26	10	15	2	1141	1133	2276	4	2
27	11	12	5	1104	903	2012	4	3
28	11	15	6	1160	949	2114	4	3
29	11	16	6	1194	1179	2379	4	3
30	12	13	4	1160	949	2112	4	3
31	12	16	6	1110	915	2030	4	3
32	13	16	2	130	75	207	4	2
33	13	17	2	36	1147	1185	4	2
34	14	15	6	1102	908	2016	4	3
35	14	19	2	1165	959	2126	4	2
36	15	16	5	1102	902	2009	4	3
37	15	21	4	1105	915	2024	4	3
38	16	17	2	1129	47	1178	4	2
39	16	23	6	1100	1000	2106	4	3
40	16	24	2	1409	1153	2563	4	2
41	17	24	6	1253	1036	2295	4	3
42	18	19	2	108	0	110	4	2
43	18	26	2	181	73	256	4	2
44	18	31	2	21	100	123	4	2
45	19	20	6	141	1414	1562	4	3
46	19	26	2	1253	1036	2291	4	2
47	20	21	6	1425	1174	2605	4	3
48	20	22	6	1204	997	2207	4	3
49	20	27	6	1175	975	2155	4	3
50	20	28	2	1297	1288	2587	4	2

TABLE 16

DISMOUNTED TROOP UNIT - RULE 1 (CONT'D.)

51	21	22	2	1107	1094	2203	4	2
52	22	23	5	1134	953	2092	4	3
53	22	29	5	1359	1345	2709	4	3
54	23	24	6	1283	1050	2338	4	3
55	23	25	6	1190	980	2175	4	3
56	23	29	6	1197	995	2198	4	3
57	24	25	5	1020	1008	2033	4	3
58	25	30	4	1174	1118	2296	4	3
59	26	27	2	1183	1174	2359	4	2
60	26	31	2	1306	474	1782	4	2
61	27	28	2	1100	1000	2102	4	2
62	27	31	5	1230	112	1347	4	3
63	27	32	6	6	101	113	4	3
64	27	33	2	138	1269	1408	4	2
65	28	29	2	1556	1414	2972	4	2
66	28	33	2	1100	1000	2102	4	2
67	28	34	2	1185	0	1187	4	2
68	29	34	6	94	1036	1136	4	3
69	30	34	6	1301	1071	2378	4	3
70	31	32	6	204	1504	1714	4	3
71	32	33	4	100	1903	2007	4	3
72	33	34	2	112	1062	1176	4	2

TABLE 17

DISMOUNTED TROOP UNIT - RULE 2

Arc	Node H	Node T	Road Width	Off-Road Left	Off-Road Right	Total Arc Width	Speed On	Speed Off
1	1	2	6	990	141	1137	4	3
2	1	8	6	0	0	6	4	3
3	2	3	6	1204	218	1428	4	3
4	2	9	2	0	0	2	4	2
5	2	10	2	1187	975	2164	4	2
6	3	4	6	1118	203	1327	4	3
7	3	10	2	1202	160	1364	4	2
8	4	5	6	1230	670	1906	4	3
9	4	10	6	1216	59	1280	4	3
10	4	11	5	0	1042	1047	4	3
11	5	6	6	1462	664	2132	4	3
12	5	11	4	1304	1078	2386	4	3
13	5	12	6	1092	1083	2181	4	3
14	6	12	6	1114	916	2036	4	3
15	7	6	4	1185	538	1727	4	3
16	7	12	4	1357	1110	2471	4	3
17	7	13	4	1115	912	2032	4	3
18	8	9	5	0	0	5	4	3
19	8	18	2	0	0	2	4	2
20	9	10	2	1230	1062	2294	4	2
21	9	14	2	1109	920	2030	4	2
22	9	18	2	0	0	2	4	2
23	9	19	2	0	0	2	4	2
24	10	11	6	1100	1000	2106	4	3
25	10	14	5	1267	1037	2309	4	3
26	10	15	2	1141	1133	2276	4	2
27	11	12	5	1104	903	2012	4	3
28	11	15	6	1160	949	2114	4	3
29	11	16	6	1194	1179	2379	4	3
30	12	13	4	1160	949	2112	4	3
31	12	16	6	1110	915	2030	4	3
32	13	16	2	0	75	77	4	2
33	13	17	2	0	1147	1149	4	2
34	14	15	6	1102	908	2016	4	3
35	14	19	2	1165	959	2126	4	2
36	15	16	5	1102	902	2009	4	3
37	15	21	4	1105	915	2024	4	3
38	16	17	2	1129	47	1178	4	2
39	16	23	6	1100	1000	2106	4	3
40	16	24	2	1409	1153	2563	4	2
41	17	24	6	1253	1036	2295	4	3
42	18	19	2	0	0	2	4	2
43	18	26	2	0	0	2	4	2
44	18	31	2	0	0	2	4	2
45	19	20	6	0	1414	1420	4	3
46	19	26	2	1253	1036	2291	4	2
47	20	21	6	1425	1174	2605	4	3
48	20	22	6	1204	997	2207	4	3
49	20	27	6	1175	975	2155	4	3
50	20	28	2	1297	1288	2587	4	2

TABLE 17

DISMOUNTED TROOP UNIT - RULE 2 (CONT'D.)

51	21	22	2	1107	1094	2203	4	2
52	22	23	5	1134	953	2092	4	3
53	22	29	5	1359	1345	2709	4	3
54	23	24	6	1283	1050	2338	4	3
55	23	25	6	1190	980	2175	4	3
56	23	29	6	1197	995	2198	4	3
57	24	25	5	1020	1008	2033	4	3
58	25	30	4	1174	1118	2296	4	3
59	26	27	2	1183	1174	2359	4	2
60	26	31	2	1306	475	1783	4	2
61	27	28	2	1100	1000	2102	4	2
62	27	31	5	1230	112	1347	4	3
63	27	32	6	0	0	6	4	3
64	27	33	2	0	1269	1271	4	2
65	28	29	2	1556	1414	2972	4	2
66	28	33	2	1100	1000	2102	4	2
67	28	34	2	1185	0	1187	4	2
68	29	34	6	0	1036	1042	4	3
69	30	34	6	1301	1071	2378	4	3
70	31	32	6	204	1504	1714	4	3
71	32	33	4	0	1903	1907	4	3
72	33	34	2	0	1062	1064	4	2

APPENDIX B
DOCUMENTATION FOR COMPUTER PROGRAM

SINGLE ARC ATTRIBUTES

********** DOCUMENTATION **********

This FORTRAN program determines the attributes for individual arc from the 100 meter grid square data, node characteristics and arc characteristics of the network overlayed on the gridded terrain.

Assumption :

- If the angle of inclination is between 45 degrees and 135 degrees, then use X-axis.
- If the angle of inclination is greater than 135 degrees or less than 45 degrees, then use the Y-axis.

FUNCTION XCORD : Compute X grid coordinates.

FUNCTION YCORD : Compute Y grid coordinates.

SUBROUTINE INIT : Set the initial conditions.

SUBROUTINE INPUT1: Read the data from data 1, data 2, and data 3.

SUBROUTINE INPUT2: Read the data about the arc.

SUBROUTINE DST : Compute distance along arc.

SUBROUTINE SLOP1 : Compute slopes between each pair of point along arc.

SUBROUTINE SLOP2 : Compute slopes between each pair of point off arc.

SUBROUTINE MIND : Compute minimum boundary distance from the arc.

SUBROUTINE FLOW : Compute flow rate for each arc.

SUBROUTINE TIME : Compute min traversal time for each arc.

SUBROUTINE PRT : Print the results of calculation.

********** VARIABLE DEFINITION **********

CASE : option of military unit type

CASE 1; vehicle unit

CASE 2; dismounted troops

PRINT : option of print

PRINT 1; print all results of calculation.

PRINT 2; print the minimum distance off arc.

PRINT 3; print the input data for checking.

PRINT 4; print the coordinates of boundary line.

PRINT 5; print the data for network.

UNITF : type of formation

UNITF 1; multiple column formation

UNITF 2; a single column formation

THRESHOLD of slope for boundary line

THRES1 : threshold for positive slope

THRES2 : threshold for negative slope

ARCTP : arc type from a head node to a tail node.

ACTD : actual surface distance of each pair of points.

AD : total actual surface distance along the arc.

ADIST : distance between head node and tail node (in kilometers).

ALTD : altitude of each point (raw data).

ALT1 : altitude of the head node on the arc.

ALT2 : altitude of the tail node on the arc.

ANGLE : angle between arc and X-axis (in degree).

AREAL : real variable by using array.

ASLPCH : average slope change along the arc.

ATIME : minimum traversal time for each arc.

AVGSLP : average slope along the arc.

DELTAX : distance between node 1 and node 2 along X-axis.

DELTAY : distance between node 1 and node 2 along Y-axis.

DIST : distance between node 1 and node 2 on the map.

DISTA : distance between each pair of point along arc.

DISTB : distance between each pair of point off arc.

DHT : difference between height of each pair of point along arc.

DHT2 : difference between height of each pair of point off arc.

DSTN : distance of each element in line (in meters).

DW : doctrinal width of battalion during deployment.

DD : doctrinal depth of battalion during deployment.

FORWV : minimum formation width of vehicle unit.

FORWD : minimum formation width of dismounted troops.

FR : flow rate of the arc (battalion / hour).

HT : height of intercept between arc and Y-axis along arc.

HT2 : height of intercept between arc and X-axis off arc.

INTS : single integer variable.

INTA : integer variable by using array.

IX,IY : crossing points between an arc and 100 meter X, Y grid line.

KNODE : number of crossing point along the arc.

KR : coefficient of route availability.

LMIN : minimum distance of the left side off arc.

MAXNO : max number of elements that move in line along the arc.

NARC : total number of arc in the data file.

NCODE : characteristics of each point (raw data).

NCODEX : dummy variable for characteristics of each point.

NODE : total number of node in the data file.

NODEA : identification number of head node on each arc.

NODEB : identification number of tail node on each arc.

NSPCL : variable of using width determination rule 2 for boundary.

NX : dummy variable for altitude of each point (raw data).

RMIN : minimum distance of the right side off arc.

ROADW : width of road itself.

SLOPE1 : slope between each pair of point along arc.

SLOPE2 : slope between each pair of point off arc.

SLPCH : slope change between each pair of point along the arc.

SREAL : single real variable.

SP : movement speed for an element of unit.

TSLPCH : total slope change along the arc.

UTMX : U.T.M. grid coordinates of the point (East - West).

UTMY : U.T.M. grid coordinates of the point (South-North).

WIDTH : width of arc (road width + off-road width).

XA,YA : intercept point between arc and Y-axis.

XB,YB : intercept point between reference line and X-axis.

XL,YL : X,Y coordinates of minimum distance of left side off arc.

XR,YR : X,Y coordinates of minimum distance of right side off arc.

XNODE : X coordinates of each node.

YNODE : Y coordinates of each node.

XM : slope between X-axis and an arc.

YM : slope between Y-axis and an arc.

CARTESIAN SPACE NETWORK

********** DOCUMENTATION **********

This FORTRAN program determines the minimum time path and minimum distance path from the undirected graph by using a shortest path algorithm. Input data of this program is output data from the single arc attributes program.

SUBROUTINE OPTION: Determine the initial options.

SUBROUTINE INIT : Set the initial conditions.

SUBROUTINE FORMTN: Specify the formation width and depth.

SUBROUTINE SHORTP: Determine the minimum time or distance path.

SUBROUTINE PRINT : Print the results of calculation.

********** VARIABLE DEFINITION **********

CASE : Option of Military Unit Type

CASE 1; vehicle unit

CASE 2; dismounted troop unit

UNITF : Type of Formation

UNITF 1; multiple column formation

UNITF 2; a single column formation

PARAM : Parameter for Shortest Path

PARAM 1; minimum time path

PARAM 2; minimum distance path

BIG : an arbitrary large real number.

DIST : distance between head node and tail node on the map.

FLOW : flow rate of each arc.

HEAD : head node of each arc.

INODE : tail node of each arc.

INTS : single integer variable.

INTA : integer variable by using array.

ISPATH : integer vector of length N.

ISTART : start node of the minimum time or distance path.

IWORK2 : integer vector of length N.

IWORK3 : integer vector of length M.

JNODE : head node of each arc.

LARC : arc number along the minimum parameter path.

LAST : terminal node of the minimum time or distance path.

LINK : arc number in the graph (link(head node, tail node)).

M : number of arcs (edges) in the graph.

MINE : arc number which is on the minefield.

N : number of nodes in the graph.

NARC : total number of arc in the data file.

NODE : total number of node in the data file.

NO : arc number in the graph (one dimensional array).

NP : the integer NP has the value zero if a shortest path is found between ISTART and LAST, otherwise it has the value one.

NUMP : the number of nodes in the shortest path found between ISTART and LAST.

PATHL : total distance between ISTART and LAST.

REALS : single real variable.

REALA : real variable by using array.

RESM : character variable RESM has the value Y if there is a mine field in the graph, otherwise it has the value N.

TAIL : tail node of each arc.

TIME : minimum traversal time of each arc.

TPATH : total traversal time for the entire unit between ISTART and LAST.

WIDTH : width of arc (road width + off-road width).

WK4 : real vector of length N.

XLEN : total value of parameter between ISTART and LAST.

APPENDIX C
COMPUTER PROGRAM FOR SINGLE ARC ATTRIBUTES

This appendix contains the computer program used to support the algorithms in Chapter III.

```
      PROGRAM  TSIM
*     JULY     5,  1987  (20:00)
*     *************  VARIABLE DECLARATION  *************

      INTEGER    ALTD,ARCTP,CASE,CODE,PRINT,UTMX,UTMY,UNITF
      REAL       LMIN
      CHARACTER  *1 BLANK
      DIMENSION  ALTD(100,100),NCODE(100,100),NX(10),NCODEX(10)
      DIMENSION  XNODE(90),YNODE(90),CODE(0:7,0:7)
      DIMENSION  DISTA(90),XA(90),YA(90)
      DIMENSION  HT(90),DHT(90),SLOPE1(90)
      DIMENSION  DISTB(100,100),XB(100,100),YB(100,100)
      DIMENSION  HT2(100,100),DHT2(100,100),SLOPE2(100,100)
      DIMENSION  NO(90),SLPCH(90)
      DIMENSION  XL(10),YL(10),XR(10),YR(10)
      DIMENSION  SP(7,3,2),FR(5,5),MAXNO(3,3)
      DIMENSION  ACTD(90),ATIME(3)
      DIMENSION  ROADW(7)
      COMMON / INTS  / NSTEP,IX1,IY1,IX2,IY2,JA,JB,K,NARC,NCOUNT,
     *                 NODEA,NODEB,J1,J2,J3,J4,PRINT,UTMX,UTMY,
     *                 CASE,ARCTP,FR,AD,UNITF,NSPCL
      COMMON / INTA  / NCODE,NO,ALTD,CODE,MAXNO
      COMMON / SREAL / XM,ANGLE,DIST,TSLPCH,ASLPCH,ALT1,ALT2,AVGSLP,
     *                 LMIN,RMIN,XL,YL,XR,YR,YM,RY,WIDTH,THRES1,THRES2
      COMMON / AREAL / XA,YA,HT,DHT,DISTA,DISTB,SLPCH,SLOPE1,SLOPE2,
     *                 XB,YB,HT2,DHT2,SP,ACTD,ATIME,ROADW
*-----------------------------------------------------------------------
* 1. SET THE INITIAL OPTIONS.
*-----------------------------------------------------------------------

      WRITE(6,601)
601   FORMAT(' ',////////)
16    PRINT*, '          WHICH TYPE OF UNIT DO YOU DESIRE? ( 1 OR 2 )'
      PRINT*, '
      PRINT*, '          *********************************'
      PRINT*, '          *                               *'
      PRINT*, '          *   1. VEHICLE UNIT             *'
      PRINT*, '          *                               *'
      PRINT*, '          *   2. DISMOUNTED TROOPS        *'
      PRINT*, '          *                               *'
      PRINT*, '          *********************************'
      READ(5,*)  CASE
      PRINT*, '** NOTE : FOR YOUR REFERENCE, CURRENT ANSWER IS',
     *         CASE
           IF ( CASE .EQ. 1 .OR. CASE .EQ. 2 ) THEN
           GO TO 17
           ELSE IF ( CASE .NE. 1 .OR. CASE .NE. 2 ) THEN
      PRINT*, '
      PRINT*, '*** ERROR : ENTER THE NUMBER 1 OR 2 ***'
      PRINT*, '
           GO TO  16
           END IF
17    WRITE(6,602)
602   FORMAT(' ',/////////)
      PRINT*, '
      PRINT*, '          WHICH TYPE OF FORMATION DO YOU DESIRE?'
```

```
      PRINT*, '                '
      PRINT*, '        ************************************'
      PRINT*, '        *                                  *'
      PRINT*, '        *   1. MULTIPLE COLUMN FORMATION   *'
      PRINT*, '        *                                  *'
      PRINT*, '        *   2. SINGLE COLUMN FORMATION     *'
      PRINT*, '        *                                  *'
      PRINT*, '        ************************************'
      READ(5,*)  UNITF
      PRINT*, '** NOTE : FOR YOUR REFERENCE, CURRENT ANSWER IS',
     *          UNITF
           IF ( UNITF .EQ. 1 .OR. UNITF .EQ. 2 ) THEN
           GO TO 18
           ELSE IF ( UNITF .NE. 1 .OR. UNITF .NE. 2 ) THEN
      PRINT*, '                '
      PRINT*, '*** ERROR : ENTER THE NUMBER 1 OR 2 ***'
      PRINT*, '                '
           GO TO  17
           END IF
18    WRITE(6,603)
603   FORMAT(' ',/////////)
      PRINT*, '
      PRINT*, '        DO YOU WANT TO USE SPECIAL RULE FOR BOUNDARY?'
      PRINT*, '          YES ( 1 ) OR NO ( 2 )'
      READ(5,*)  NSPCL
      PRINT*, '** NOTE : FOR YOUR REFERENCE, CURRENT ANSWER IS',
     *            NSPCL
      WRITE(6,604)
604   FORMAT(' ',////////////)
      PRINT*, '                '
      PRINT*, '        WHICH OUTPUT DO YOU WISH TO PRINT? ( 1 - 5 )'
      PRINT*, '                '
      PRINT*, '        ******************************************'
      PRINT*, '        *   1. GENERAL INFORMATION FOR EACH ARC  *'
      PRINT*, '        *   2. FLOW RATE & TRAVERSAL TIME        *'
      PRINT*, '        *   3. INPUT DATA FOR CHECK              *'
      PRINT*, '        *   4. COORDINATES OF BOUNDARY LINE      *'
      PRINT*, '        *   5. NETWORK DATA FOR NETWORK PROGRAM  *'
      PRINT*, '        ******************************************'
      READ(5,*)  PRINT
      PRINT*, '** NOTE : FOR YOUR REFERENCE, CURRENT ANSWER IS',
     *            PRINT
           IF ( PRINT .EQ. 1 .OR. PRINT .EQ. 2 .OR. PRINT .EQ. 3
     *       .OR. PRINT .EQ. 4 .OR. PRINT .EQ. 5 ) THEN
           GO TO 19
           ELSE IF ( PRINT .NE. 1 .OR. PRINT .NE. 2 .OR. PRINT .NE. 3
     *       .OR. PRINT .NE. 4 .OR. PRINT .NE. 5 ) THEN
      PRINT*, '                '
      PRINT*, '*** ERROR : ENTER THE NUMBER 1 THRU 5 ***'
      PRINT*, '                '
           GO TO  18
           END IF
*     THRESHOLD FOR BARRIER DUE TO SLOPE OFF ROAD
19         THRES1 =  0.3
           THRES2 = -0.3
*-----------------------------------------------------------------------
* 2. SET THE INITIAL CONDITIONS.
*-----------------------------------------------------------------------
*                          TOTAL NUMBER OF ARCS IN NETWORK.
           NARC   =  72
*                          TOTAL NUMBER OF NODES IN NETWORK.
           NODE   =  34
      CALL INIT(NX,NCODEX,XL,YL,XR,YR,XA,YA,SLPCH,HT,NO,DHT,CASE,
     *          DISTA,SLOPE1,XB,YB,HT2,DHT2,DISTB,SLOPE2,CODE)
*-----------------------------------------------------------------------
* 3. READ THE DATA FROM THE DATA FILE.(DATA 1, DATA 2, DATA 3, DATA 4)
*-----------------------------------------------------------------------
```

```
*              DATA 1 ; ALTITUDE  AND  CODE
*              DATA 2 ; NODE CHARACTERISTICS
*              DATA 3 ; ARC  CHARACTERISTICS
*              DATA 4 ; SPEED  FOR  MOVEMENT
         CALL INPUT1(NARC,NODE,ALTD,NCODE,XNODE,YNODE,
     *                UNITF,CASE,THRES1,THRES2,SP,PRINT)
              NCOUNT  = 1
              IY      = 100
         DO   10  IB  = 1, NARC
         CALL INPUT2(XNODE,YNODE,IY,IX1,IY1,IX2,IY2,NODEA,NODEB,
     *                ARCTP,UTMX,UTMY)
*-----------------------------------------------------------------
* 4. DETERMINE THE ANGLE BETWEEN ARC AND X-AXIS.
*-----------------------------------------------------------------

              DELTAX =  IX2 - IX1
              DELTAY =  IY2 - IY1
                   IF ( DELTAX .NE. 0.) THEN
                        XM     =  DELTAY / DELTAX
                   ELSE IF ( DELTAX .EQ. 0.) THEN
                        XM   =   9999999.
                   END  IF
              RADIAN =  ATAN( XM )
              DIST   =  SQRT( DELTAX ** 2 + DELTAY ** 2 )
              ANGLE1 =  ( RADIAN * 180.) / 3.141592
                   IF ( ANGLE1 .GE. 0.) THEN
                        ANGLE  =  ANGLE1
                   ELSE IF ( ANGLE1 .LT. 0.) THEN
                        ANGLE  = 180. + ANGLE1
                   END  IF
*-----------------------------------------------------------------
* 5. COMPUTE THE DISTANCE BETWEEN EACH PAIR OF POINTS ALONG ARC.
*-----------------------------------------------------------------

         CALL DST(IY1,IX1,IY2,IX2,ANGLE,DIST,XM,XA,YA,DISTA,K)
*-----------------------------------------------------------------
* 6. COMPUTE AVERAGE SLOPE ALONG THE ARC.
*-----------------------------------------------------------------

              LX1 = IX1 / 100
              LY1 = IY1 / 100
              LX2 = IX2 / 100
              LY2 = IY2 / 100
                 ALT1   =  ALTD( LX1, LY1 )
                 ALT2   =  ALTD( LX2, LY2 )
                 AVGSLP = ( ALT2 - ALT1 ) / DIST
*-----------------------------------------------------------------
* 7. COMPUTE SLOPES BETWEEN EACH PAIR OF POINTS ALONG THE ARC.
*-----------------------------------------------------------------

         CALL SLOP1(ANGLE,ALT1,ALT2,IX1,IY1,IY2,IX2,DISTA,XM,XA,YA,ALTD,
     *                HT,DHT,SLOPE1,NA,ACTD,AD)
*-----------------------------------------------------------------
* 8. AS A MEASURE OF TERRAIN " STEEPNESS " ALONG THE ARC, COMPUTE
*    TOTAL AND AVERAGE SLOPE CHANGE.
*-----------------------------------------------------------------

              ASLPCH = 0.
              TSLPCH = 0.
              NB = NA - 1
       DO  11 ID = 1, NB
              SLPCH(ID) = ABS( SLOPE1(ID+1) - SLOPE1(ID))
              TSLPCH = TSLPCH + SLPCH(ID)
11     CONTINUE
```

```
              ASLPCH  = TSLPCH / NB
*-------------------------------------------------------------------------
* 9. COMPUTE SLOPES FROM ARC TO POINT OFF ARC, AND MIN DISTANCE.
*-------------------------------------------------------------------------
       CALL  SLOP2
*-------------------------------------------------------------------------
* 10. DETERMINE COORDINATES FOR THE MIN DISTANCE.( LEFT AND RIGHT )
*-------------------------------------------------------------------------
       CALL MIND(K,XM,YM,XA,YA,LMIN,RMIN,XL,YL,XR,YR)
*-------------------------------------------------------------------------
* 11. COMPUTE  THE RATE OF FLOW ALONG THE ARC.
*-------------------------------------------------------------------------
       CALL FLOW(SP,WIDTH,ARCTP,CASE,FR,MAXNO)
*-------------------------------------------------------------------------
* 12. COMPUTE  THE MIN TRAVERSAL TIME ALONG THE ARC.
*-------------------------------------------------------------------------
       CALL TIME(ARCTP,SP,UNITF,CASE,DIST,WIDTH,ATIME)
*-------------------------------------------------------------------------
* 13. PRINT THE RESULT OF CALCULATION.
*-------------------------------------------------------------------------
       CALL   PRT
       NCOUNT = NCOUNT + 1
10     CONTINUE
       STOP
       END
*************************************************************************
* A. SUBROUTINE FOR SETING THE INITIAL CONDITIONS.
*************************************************************************
       SUBROUTINE INIT(NX,NCODEX,XL,YL,XR,YR,XA,YA,SLPCH,HT,NO,DHT,CASE,
     *                 DISTA,SLOPE1,XB,YB,HT2,DHT2,DISTB,SLOPE2,CODE)

       INTEGER    CASE,CODE
       DIMENSION  XL(10),YL(10),XR(10),YR(10),NX(10),NCODEX(10)
       DIMENSION  DISTA(90),XA(90),YA(90)
       DIMENSION  HT(90),DHT(90),SLOPE1(90)
       DIMENSION  DISTB(100,100),XB(100,100),YB(100,100)
       DIMENSION  HT2(100,100),DHT2(100,100),SLOPE2(100,100)
       DIMENSION  NO(90),SLPCH(90)
       DIMENSION  CODE(0:7,0:7)

*   INITIALIZATION FOR VARIABLES.
       DO  10  L   = 1, 10
               XL(L)  = 0.
               YL(L)  = 0.
               XR(L)  = 0.
               YR(L)  = 0.
               NX(L)  = 0
               NCODEX(L) = 0
10     CONTINUE
       DO  11  LA  = 1, 90
               XA(LA)  = 0.
               YA(LA)  = 0.
               HT(LA)  = 0.
               NO(LA)  = 0
               DHT(LA) = 0.
               SLPCH (LA) = 0.
               DISTA (LA) = 0.
```

```
              SLOPE1(LA) = 0.
11      CONTINUE
        DO  12 LB = 1, 100
        DO  12 LC = 1, 100
               XB(LB,LC)     = 0.
               YB(LB,LC)     = 0.
               HT2 (LB,LC)   = 0.
               DHT2(LB,LC)   = 0.
               DISTB (LB,LC) = 0.
               SLOPE2(LB,LC) = 0.
12      CONTINUE

*    SET CODE OF BOUNDARY FOR VEHICLE UNIT.
        IF ( CASE .EQ.  1) THEN
           DO  13  I  = 0,7
           DO  13  J  = 0,7
               CODE(I,J) = 0
13         CONTINUE
               CODE(4,4) = 1
               CODE(4,7) = 1
               CODE(7,4) = 1
               CODE(7,7) = 1
*    SET CODE OF BOUNDARY FOR DISMOUNTED TROOPS.
        ELSE IF ( CASE .EQ. 2) THEN
           DO  14  I  = 0,7
           DO  14  J  = 0,7
               CODE(I,J) = 1
14         CONTINUE
           DO  15  IA = 0,7
               L = 0
               CODE(IA,L) = 0
               CODE(IA,3) = 0
               CODE(IA,5) = 0
15         CONTINUE
        END IF
        RETURN
        END
*************************************************************************

* B. SUBROUTINE FOR READING THE DATA FILES( DATA 1, DATA 2, DATA 4 )

*************************************************************************

      SUBROUTINE INPUT1(NARC,NODE,ALTD,NCODE,XNODE,YNODE,
     *                  UNITF,CASE,THRES1,THRES2,SP,PRINT)

        INTEGER    ALTD,CASE,PRINT,UNITF
        CHARACTER  *1 BLANK
        DIMENSION  ALTD(100,100),NCODE(100,100)
        DIMENSION  NX(10),NCODEX(10),XNODE(90),YNODE(90)
        DIMENSION  SP(7,3,2)

*     READ THE ALTITUDE AND CODE BY TWO DIMENSIONAL ARRAY.
               J = 0
       DO  20  IN = 1,99
               I = 1
               J = J + 1
          DO   21 IT = 1,10
               READ(1,100)  ( NX(I1), NCODEX(I1), I1=1,10 )
100            FORMAT(I4,I2,9(I5,I2))
               DO  22  IS = 1,10
                   ALTD(I,J) = NX(IS)
                   NCODE(I,J) = NCODEX(IS)
                   I = I + 1
22             CONTINUE
21        CONTINUE
               READ(1,101) BLANK
101            FORMAT(A1)
20      CONTINUE
*      READ THE NODE CHARACTERISTICS FOR SINGLE ARC ATTRIBUTES.
```

```
          DO  23  IA = 1,NODE
              READ(2,110) XNODE(IA),YNODE(IA)
110           FORMAT( 4X,2F4.0)
23        CONTINUE
           DO  26  IM = 1,7
           DO  26  JM = 1,3
           DO  26  KM = 1,2
*     READ THE SPEED FOR MOVEMENT BY THREE DIMENSIONAL ARRAY.
              READ(4,130) SP(IM,JM,KM)
130           FORMAT(F5.0)
26         CONTINUE

       IF ( PRINT .EQ. 1 ) THEN
              WRITE(6 ,716)CASE,UNITF,THRES1,THRES2
716          FORMAT(' ',4X,'UNIT  TYPE      :',I4,/,5X,'UNIT FORMATION',
     *        '  :',I4 ,/,5X,'POS. THRESHOLD  :', F7.3,
     *        /,5X,'NEG. THRESHOLD  :',F7.3 )

       ELSE IF ( PRINT .EQ. 2 ) THEN
              WRITE(20,769)CASE,UNITF,THRES1,THRES2
769          FORMAT(' ',4X,'UNIT  TYPE      :',I4,/,5X,'UNIT FORMATION',
     *        '  :',I4 ,/,5X,'POS. THRESHOLD  :', F7.3,
     *        /,5X,'NEG. THRESHOLD  :',F7.3 )

       ELSE IF ( PRINT .EQ. 3 ) THEN
              WRITE(6,710)
710           FORMAT(' ',3X,71('-'),//,4X,' INPUT  DATA ',
     *        '( ALTITUDE   AND   CHARACTERISTICS )',/)
              NK = 1
           DO  24  NA  = 1,2
           DO  24  NB  = 1,100
              WRITE(6,711) NK,NB,NA, ALTD(NB,NA),NCODE(NB,NA)
711           FORMAT(' ',I5,'  (', I3,I3,'  ) ',I5,I5)
              NK = NK + 1
24         CONTINUE
              WRITE(6,712)
712           FORMAT(' ',3X,//,4X,' NODES ALONG ARC ',
     *        //, 9X,'NO       X       Y')
              WRITE(6,713) (XNODE(IA),YNODE(IA), IA=1,NODE)
713           FORMAT(' ',6X, I4,3X,2F7.0)

       ELSE IF ( PRINT .EQ. 5 ) THEN
       WRITE(6,714)
714    FORMAT(' ',6X,'HEAD  TAIL',3X,'TIME FLOW RATE DIST WIDTH SPEED')
       END IF
       RETURN
       END
*************************************************************************

* C. SUBROUTINE FOR READING THE DATA FILE ( DATA 3 )

*************************************************************************

      SUBROUTINE INPUT2(XNODE,YNODE,IY,IX1,IY1,IX2,IY2,NODEA,NODEB,
     *                  ARCTP,UTMX,UTMY)

        INTEGER    ARCTP,UTMX,UTMY
        DIMENSION  XNODE(90),YNODE(90)

*     READ THE ARC CHARACTERISTICS FOR SINGLE ARC ATTRIBUTES.
          READ(3,120) NODEA,NODEB,ARCTP
120       FORMAT(I3,I4,I4)
               NX1 = XNODE( NODEA )
               NY1 = YNODE( NODEA )
               UTMX= NX1 / 100
               UTMY= NY1 / 100
               UTMX= UTMX * 10
               UTMY= UTMY * 10
                     MX1 = MOD( NX1,IY ) * 100
                     MY1 = MOD( NY1,IY ) * 100
```

```
              NX2 = XNODE( NODEB )
              NY2 = YNODE( NODEB )
              MX2 = MOD( NX2,IY ) * 100
              MY2 = MOD( NY2,IY ) * 100
       IF ( MY2 .GT. MY1) THEN
              IX1 = MX1
              IY1 = MY1
              IX2 = MX2
              IY2 = MY2
*     CHANGE THE ORDER OF NODE FOR USING ANGLES OF INCLINATION.
       ELSE IF ( MY2 .LT. MY1) THEN
              IX1 = MX2
              IY1 = MY2
              IX2 = MX1
              IY2 = MY1
              NODE1 = NODEA
              NODE2 = NODEB
              NODEA = NODE2
              NODEB = NODE1
       ELSE IF ( MY2 .EQ. MY1) THEN
            IF ( MX2 .GE. MX1) THEN
              IX1 = MX1
              IY1 = MY1
              IX2 = MX2
              IY2 = MY2
            ELSE IF ( MX2 .LT. MX1) THEN
              IX1 = MX2
              IY1 = MY2
              IX2 = MX1
              IY2 = MY1
              NODE1 = NODEA
              NODE2 = NODEB
              NODEA = NODE2
              NODEB = NODE1
            END  IF
       END  IF
       RETURN
       END
*************************************************************************

* D. FUNCTION FOR CALCULATING GRID COORDINATES( X ).

*************************************************************************

       FUNCTION XCORD(XM,IA,IY,IX)

           IF ( XM .NE. 0.) THEN
              XCORD = (( 1./ XM ) * ( IA - IY ))+ IX
           ELSE IF ( XM .EQ. 0.) THEN
              XCORD =  IX
           END  IF
       RETURN
       END
*************************************************************************

* E. FUNCTION FOR CALCULATING GRID COORDINATES( Y ).

*************************************************************************

       FUNCTION YCORD(YM,XXB,YC,XC)

       YCORD = ( YM * ( XXB  - XC ))+ YC
       RETURN
       END
*************************************************************************

* F. SUBROUTINE  FOR  COMPUTING  THE  DISTANCE  BETWEEN  EACH  PAIR
*    OF POINTS ALONG ARC.

*************************************************************************
```

```
      SUBROUTINE DST(IY1,IX1,IY2,IX2,ANGLE,DIST,XM,XA,YA,
     *               DISTA,K)

       DIMENSION  DISTA(90),XA(90),YA(90)

         IF ( ANGLE .GE. 45. .AND. ANGLE .LE. 135. ) THEN
              NA    = ((IY2 - IY1 + 49 ) / 100) + 1
         ELSE IF ( ANGLE .GT. 135. .OR. ANGLE .LT. 45. ) THEN
              NA45  = ((IX2 - IX1 + 49 ) / 100) + 1
              NA135 = ((IX1 - IX2 + 49 ) / 100) + 3
              IF( ANGLE .LT.  45.) NA = NA45
              IF( ANGLE .GT. 135.) NA = NA135
         END  IF
         K     = 1
         YA(K) = IY1
         XA(K) = IX1
*     ANGLE OF INCLINATION IS BETWEEN 45 AND 135 DEGREES.
      IF ( ANGLE .GE. 45. .AND. ANGLE .LE. 135. ) THEN
           DO  30 IA = 0, IY2-100, 100
               A  = IA - IY1
               IF (A .GT. 0.) THEN
                   K  = K + 1
                   IY = IY1
                   IX = IX1
                   YA(K) = REAL( IA )
                   XA(K) = XCORD( XM,IA,IY,IX )
               END IF
30         CONTINUE
*     ANGLE OF INCLINATION IS GREATER THAN 135 DEGREES.
      ELSE IF ( ANGLE .GT. 135. ) THEN
           I1 = ( IX1 + 100 ) / 100
           I2 = ( IX2 / 100 ) + 1
           DO   31 IA = I1,I2,-1
                   K  = K + 1
                   XA(K) = IA * 100.
                   YC = IY1
                   XC = IX1
                   YM = XM
                   XXB = XA(K)
                   YA(K) = YCORD( YM, XXB, YC, XC )
31         CONTINUE
*     ANGLE OF INCLINATION IS LESS THAN 45 DEGREES.
      ELSE IF ( ANGLE .LT. 45. ) THEN
           I1 = ( IX1 + 100 ) / 100
           I2 = ( IX2  / 100 ) - 1
           DO   32 IA = I1, I2, 1
                   K  = K + 1
                   XA(K) = IA * 100.
                   YC = IY1
                   XC = IX1
                   YM = XM
                   XXB = XA(K)
                   YA(K) = YCORD( YM, XXB, YC, XC )
32         CONTINUE
       END IF

*     COMPUTE THE DISTANCE BETWEEN EACH PAIR OF POINTS.
          K = K + 1
          IY = IY1
          IX = IX1
          IA = IY2
          IF( XM .NE. 0. ) THEN
                YA(K) = IY2
                XA(K) = XCORD( XM, IA, IY, IX )
          ELSE  IF( XM .EQ. 0. ) THEN
                YA(K) = IY2
                XA(K) = IX2
          END  IF
          DIST1 =  0.
```

```
      DO   33 IB = 1, NA
           IF( IB .LT. NA ) THEN
               DISTA(IB) = SQRT(( XA(IB+1) - XA(IB)) **2
     *                         + ( YA(IB+1) - YA(IB)) **2 )
           ELSE IF( IB .EQ. NA ) THEN
               DISTA(IB) = DIST - DIST1
           END IF
           DIST1 = DIST1 + DISTA(IB)
33    CONTINUE
      RETURN
      END
*****************************************************************************

* G. SUBROUTINE FOR COMPUTING  SLOPES  BETWEEN EACH PAIR OF POINTS
*    ALONG THE ARC.

*****************************************************************************

      SUBROUTINE SLOP1(ANGLE,ALT1,ALT2,IX1,IY1,IY2,IX2,DISTA,XM,XA,YA,
     *                 ALTD,HT,DHT,SLOPE1,NA,ACTD,AD)

      INTEGER    ALTD
      DIMENSION  ALTD(100,100),DISTA(90),XA(90),YA(90)
      DIMENSION  HT(90),DHT(90),SLOPE1(90)
      DIMENSION  ACTD(90)

      AD    =   0.
      RY    = 100.
      HT(1) = ALT1
      IF ( ANGLE .GE. 45. .AND. ANGLE .LE. 135. ) THEN
           NA    = ((IY2 - IY1 + 49 ) / 100) + 1
        DO  40 IC = 2, NA
           JX1 = INT( XA(IC) / 100 )
           JY1 = INT( YA(IC) / 100 )
           R  = MOD(XA(IC),RY)
           HTA    = (R/100.) * ( ALTD(JX1,JY1) - ALTD(JX1+1,JY1))
               IF( IC .LT. NA ) THEN
                   HT(IC ) = ALTD(JX1,JY1) - HTA
               ELSE IF( IC .EQ. NA ) THEN
                   HT(IC ) = ALT2
               END IF
           DHT(IC-1)     =  HT(IC )- HT(IC-1)
           SLOPE1(IC-1) =  DHT(IC-1) / DISTA(IC-1)
           ACTD(IC-1)    =  DISTA(IC-1) / COS(SLOPE1(IC-1))
           AD = AD + ACTD(IC-1)
40    CONTINUE
      ELSE IF ( ANGLE .GT. 135. .OR. ANGLE .LT. 45. ) THEN
           NA45  = ((IX2 - IX1+ 49 ) / 100) + 1
           NA135 = ((IX1 - IX2+ 49 ) / 100) + 3
           IF( ANGLE .LT.  45.) NA = NA45
           IF( ANGLE .GT. 135.) NA = NA135
        DO   41 IC = 2, NA
           JX2 = XA(IC)  / 100
           JY2 = YA(IC)  / 100
           R  = MOD(YA(IC),RY)
           HTA    = (R/100.) * ( ALTD(JX2,JY2) - ALTD(JX2,JY2+1))
                IF( IC .LT. NA ) THEN
                    HT(IC) = ALTD(JX2,JY2) - HTA
                ELSE IF( IC .EQ. NA ) THEN
                    HT(IC) = ALT2
                END  IF
           DHT(IC-1)     =  HT(IC)- HT(IC-1)
           SLOPE1(IC-1) =  DHT(IC-1) / DISTA(IC-1)
           ACTD(IC-1)    =  DISTA(IC-1) / COS(SLOPE1(IC-1))
           AD = AD + ACTD(IC-1)
41    CONTINUE
      END IF
      RETURN
      END
*****************************************************************************
```

```
* H. SUBROUTINE FOR COMPUTING  SLOPES FROM ARC TO POINTS OFF ARC.
*************************************************************************

      SUBROUTINE  SLOP2

       INTEGER     ALTD,ARCTP,CASE,CODE,PRINT,UTMX,UTMY
       REAL        LMIN
       DIMENSION   ALTD(100,100),NCODE(100,100),NX(10),NCODEX(10)
       DIMENSION   XNODE(90),YNODE(90),CODE(0:7,0:7)
       DIMENSION   DISTA(90),XA(90),YA(90)
       DIMENSION   HT(90),DHT(90),SLOPE1(90)
       DIMENSION   DISTB(100,100),XB(100,100),YB(100,100)
       DIMENSION   HT2(100,100),DHT2(100,100),SLOPE2(100,100)
       DIMENSION   NO(90),SLPCH(90)
       DIMENSION   XL(10),YL(10),XR(10),YR(10)
       DIMENSION   SP(7,3,2),FR(5,5),MAXNO(3,3)
       DIMENSION   ACTD(90),ATIME(3)
       DIMENSION   ROADW(7)

       COMMON / INTS   / NSTEP,IX1,IY1,IX2,IY2,JA,JB,K,NARC,NCOUNT,
     *                   NODEA,NODEB,J1,J2,J3,J4,PRINT,UTMX,UTMY,
     *                   CASE,ARCTP,FR,AD,UNITF,NSPCL
       COMMON / INTA   / NCODE,NO,ALTD,CODE,MAXNO
       COMMON / SREAL  / XM,ANGLE,DIST,TSLPCH,ASLPCH,ALT1,ALT2,AVGSLP,
     *                   LMIN,RMIN,XL,YL,XR,YR,YM,RY,WIDTH,THRES1,THRES2
       COMMON / AREAL  / XA,YA,HT,DHT,DISTA,DISTB,SLPCH,SLOPE1,SLOPE2,
     *                   XB,YB,HT2,DHT2,SP,ACTD,ATIME,ROADW

              LMIN = 10000.
              RMIN = 10000.
              RY   =   100.
                   IF ( XM .NE. 0.) THEN
                        YM  = -  1. / XM
                   ELSE IF ( XM .EQ. 0.) THEN
                        YM  =  9999999.
                   END  IF

*     ANGLE OF INCLINATION IS BETWEEN 45 AND 135 DEGREES.
       IF ( ANGLE .GE. 45. .AND. ANGLE .LE. 135. ) THEN

*     COMPUTE THE SLOPE ON THE LEFT SIDE.
      DO  70  IE  = 1,K
              KD  = 1
              IH  = (INT(XA(IE) / 100) * 100)- 100
              IH1 = IH
72            XB(IE,KD) =  IH
              YC  = YA(IE)
              XC  = XA(IE)
              XXB = XB(IE,KD)
              YB(IE,KD) =  YCORD( YM, XXB, YC, XC)
              J1  = XB(IE,KD) / 100
              J2  = YB(IE,KD) / 100
              RA  = (MOD( YB(IE,KD), RY)) / 100.
                  IF ( XM .GE. 0. ) THEN
                       IF( RA .GE. 0.5 ) J4 = J2 + 1
                       IF( RA .LT. 0.5 ) J4 = J2
                  ELSE IF ( XM  .LT. 0. ) THEN
                       IF( RA .GE. 0.5 ) THEN
                             J4 = J2
                       ELSE IF( RB .LT. 0.5 ) THEN
                             J4 = J2 - 1
                       END IF
                  END  IF
         IF ( XB(IE,KD) .GT. 10000. .OR. YB(IE,KD) .GE. 10000. ) THEN
              KD = KD -1
              GO TO 75
         END  IF
              DISTB(IE,KD) =  SQRT(( XB(IE,KD) - XA(IE)) **2
```

```
     *                              + ( YB(IE,KD) - YA(IE)) **2)
               HTB =  RA  * (ALTD(J1,J2) - ALTD(J1,J2+1))
               HT2(IE,KD)  = ALTD(J1,J2) - HTB
               DHT2(IE,KD) = HT2(IE,KD) - HT(IE)
               IF( DISTB(IE,KD) .EQ. 0.) GO TO 70
               SLOPE2(IE,KD) = DHT2(IE,KD) / DISTB(IE,KD)
                    IF( KD .EQ. 1 ) THEN
                         NCA = 7
                    ELSE IF( KD .GT. 1 ) THEN
                         JA = XB(IE,KD-1) / 100
                         JB = YB(IE,KD-1) / 100
                         NCA = NCODE( JA,JB)
                    END IF
               NCB = NCODE(J1,J4)
75        IF ( CODE(NCA,NCB) .EQ. 1 .AND. SLOPE2(IE,KD) .LE. THRES1
     *         .AND. SLOPE2(IE,KD) .GE. THRES2 .AND. KD .LE. 10) THEN
               IH = IH - 100
               KD = KD + 1
               GO TO 72
          ELSE IF (CODE( NCA,NCB ) .NE. 1 .OR. SLOPE2(IE,KD) .GT. THRES1
     *         .OR. SLOPE2(IE,KD) .LT. THRES2  .OR. KD .GT. 10 ) THEN
               IH = IH1 + 200
               IF(DISTB(IE,KD) .LT. LMIN) LMIN = DISTB(IE,KD)
     *    APPLY SPECIAL RULE(WIDTH DETERMINATION RULE 2) FOR BOUNDARY.
               IF( NSPCL .EQ. 1 ) THEN
                      IF( KD .EQ. 1 ) THEN
                             LMIN = 0.
                      END  IF
               END  IF
               GO TO 71
          END  IF
     *    COMPUTE THE SLOPE ON THE RIGHT SIDE.
     71        KD = KD + 1
     74        XB(IE,KD) =  IH
               YC  = YA(IE)
               XC  = XA(IE)
               XXB = XB(IE,KD)
               YB(IE,KD) =  YCORD( YM, XXB, YC, XC )
               J1  = XB(IE,KD) / 100
               J2  = YB(IE,KD) / 100
               RB  = (MOD (YB(IE,KD), RY)) / 100.
               D2  = ABS( YA(IE) - YB(IE,KD) )
                    IF ( XM  .GE. 0. ) THEN
                         IF( RB .GE. 0.5 ) J4 = J2 + 1
                         IF( RB .LT. 0.5 .AND. D2 .GT. 50.) J4 = J2
                    ELSE IF ( XM  .LT. 0. ) THEN
                         IF( RB .GE. 0.5 ) J4 = J2 + 1
                         IF( RB .LT. 0.5 ) J4 = J2
                    END  IF
              IF(XB(IE,KD) .GT. 10000. .OR. YB(IE,KD) .GE. 10000. ) THEN
                    IF(DISTB(IE,KD-1) .LT. RMIN) THEN
                         RMIN = DISTB(IE,KD-1)
                    END  IF
                    GO TO 77
               END   IF
                 DISTB(IE,KD) =  SQRT(( XB(IE,KD) - XA(IE)) **2
     *                              + ( YB(IE,KD) - YA(IE)) **2)
                 HTB =  RB * (ALTD(J1,J2) - ALTD(J1,J2+1))
                 HT2(IE,KD)  = ALTD(J1,J2) - HTB
                 DHT2(IE,KD) = HT2(IE,KD) - HT(IE)
                 IF( DISTB(IE,KD) .EQ. 0.) GO TO 76
                 SLOPE2(IE,KD) = DHT2(IE,KD) / DISTB(IE,KD)
                    IF ( IH .EQ. IH1+200 ) THEN
                         NCA = 7
                    ELSE IF( KD .GT. 1 ) THEN
                         JA = XB(IE,KD-1) / 100
                         JB = YB(IE,KD-1) / 100
                         NCA = NCODE( JA,JB)
                    END  IF
                 NCB = NCODE(J1,J4)
```

```
76        IF ( CODE(NCA,NCB) .EQ. 1 .AND. SLOPE2(IE,KD) .LE. THRES1
     *       .AND. SLOPE2(IE,KD) .GE. THRES2 .AND. KD .LE. 20) THEN
               IH = IH + 100
               KD = KD + 1
               GO TO 74
          ELSE IF (CODE(NCA,NCB) .NE. 1 .OR. SLOPE2(IE,KD) .GT. THRES1
     *       .OR. SLOPE2(IE,KD) .LT. THRES2 .OR. KD .GT. 20) THEN
               IF(DISTB(IE,KD) .LT. RMIN) THEN
                    RMIN = DISTB(IE,KD)
               END  IF
*    APPLY SPECIAL RULE(WIDTH DETERMINATION RULE 2) FOR BOUNDARY.
               IF( NSPCL .EQ. 1 ) THEN
                     IF( KD .EQ. 2 ) THEN
                          RMIN = 0.
                     END  IF
               END  IF
               KD = KD + 1
          END  IF
77        NO(IE) = KD - 1
70      CONTINUE

*      ANGLE OF INCLINATION IS GREATER THAN 135 OR LESS THAN 45 DEGREES.
       ELSE IF ( ANGLE .GT. 135. .OR. ANGLE .LT. 45.) THEN

*     COMPUTE THE SLOPE ON THE LEFT SIDE.
       DO  60  IE  = 1,K
               KD  = 1
               IF( ANGLE .GT. 135.) IH  = (INT(YA(IE)/100) * 100) - 100
               IF( ANGLE .LT.  45.) IH  = (INT(YA(IE)/100) * 100) + 100
                    IH1 = IH
62             YB(IE,KD) =  IH
               IY  = YA(IE)
               IX  = XA(IE)
               IA  = YB(IE,KD)
               XB(IE,KD) =  XCORD( YM, IA, IY, IX )
                 IF( XB(IE,KD) .LE. 0.) THEN
                      IH = IH1 + 200
                      GO TO 64
                 END IF
               J1  = XB(IE,KD) / 100
               J2  = YB(IE,KD) / 100
               RA  = (MOD( XB(IE,KD), RY)) / 100.
                 IF( RA .GE. 0.5 ) THEN
                      J3 = J1 + 1
                 ELSE IF( RA .LT. 0.5 ) THEN
                      J3 = J1
                 END IF
          IF ( XB(IE,KD) .GT. 10000. .OR. YB(IE,KD) .GE. 10000. ) THEN
               KD = KD -1
               GO TO 65
          END  IF
               DISTB(IE,KD) =  SQRT(( XB(IE,KD) - XA(IE)) **2
     *                              + ( YB(IE,KD) - YA(IE)) **2)
               HTB =  RA  * (ALTD(J1,J2) - ALTD(J1+1,J2))
               HT2(IE,KD)  = ALTD(J1,J2) - HTB
               DHT2(IE,KD) = HT2(IE,KD) - HT(IE)
               IF( DISTB(IE,KD) .EQ. 0.) GO TO 65
               SLOPE2(IE,KD) = DHT2(IE,KD) / DISTB(IE,KD)
                    IF( KD .EQ. 1 ) THEN
                         NCA = 7
                    ELSE IF( KD .GT. 1 ) THEN
                         JA = XB(IE,KD-1) / 100
                         JB = YB(IE,KD-1) / 100
                         NCA = NCODE( JA,JB)
                    END IF
               NCB = NCODE(J1,J4)
65       IF( ANGLE .GT. 135.) THEN
            IF ( CODE(NCA,NCB) .EQ. 1 .AND. SLOPE2(IE,KD) .LE. THRES1
     *          .AND. SLOPE2(IE,KD) .GE. THRES2 .AND. KD .LE. 10) THEN
                 IH = IH - 100
```

```
                KD = KD + 1
                GO TO 62
          ELSE IF (CODE(NCA,NCB) .NE. 1 .OR. SLOPE2(IE,KD) .GT. THRES1
     *          .OR. SLOPE2(IE,KD) .LT. THRES2 .OR. KD .GT. 10) THEN
                IH = IH1 + 200
                IF(DISTB(IE,KD) .LT. LMIN) THEN
                     LMIN = DISTB(IE,KD)
                END  IF
     *    APPLY SPECIAL RULE(WIDTH DETERMINATION RULE 2) FOR BOUNDARY.
                IF( NSPCL .EQ. 1 ) THEN
                     IF( KD .EQ. 1 ) THEN
                          LMIN = 0.
                     END  IF
                END  IF
                GO TO 61
          END  IF
        ELSE   IF( ANGLE .LT.  45.) THEN
          IF ( CODE(NCA,NCB) .EQ. 1 .AND. SLOPE2(IE,KD) .LE. THRES1
     *          .AND. SLOPE2(IE,KD) .GE. THRES2 .AND. KD .LE. 10) THEN
                IH = IH + 100
                KD = KD + 1
                GO TO 62
          ELSE IF(CODE(NCA,NCB) .NE. 1 .OR. SLOPE2(IE,KD) .GT. THRES1
     *          .OR. SLOPE2(IE,KD) .LT. THRES2 .OR. KD .GT. 10) THEN
                IH = IH1 - 200
                IF(DISTB(IE,KD) .LT. LMIN) THEN
                     LMIN = DISTB(IE,KD)
                END  IF
     *    APPLY SPECIAL RULE(WIDTH DETERMINATION RULE 2) FOR BOUNDARY.
                IF( NSPCL .EQ. 1 ) THEN
                     IF( KD .EQ. 1 ) THEN
                          LMIN = 0.
                     END  IF
                END  IF
                GO TO 61
           END IF .
        END IF

     *    COMPUTE THE SLOPE ON THE RIGHT SIDE.
61              KD = KD + 1
64              YB(IE,KD) =  IH
                IY  = YA(IE)
                IX  = XA(IE)
                IA  = YB(IE,KD)
                XB(IE,KD) =  XCORD( YM, IA, IY, IX)
                J1  = XB(IE,KD) / 100
                J2  = YB(IE,KD) / 100
                RB  = (MOD (XB(IE,KD), RY)) / 100.
                    IF( RB .GE. 0.5 ) THEN
                        J3 = J1 + 1
                    ELSE IF( RB .LT. 0.5 ) THEN
                        J3 = J1
                    END IF
             IF(XB(IE,KD) .GT. 10000. .OR. YB(IE,KD) .GE. 10000. ) THEN
                    IF(DISTB(IE,KD-1) .LT. RMIN) THEN
                        RMIN = DISTB(IE,KD-1)
                    END IF
                    GO TO 67
             END  IF
               DISTB(IE,KD) =  SQRT(( XB(IE,KD) - XA(IE)) **2
     *                             + ( YB(IE,KD) - YA(IE)) **2)
               HTB =  RB * (ALTD(J1,J2) - ALTD(J1+1,J2))
               HT2(IE,KD)  = ALTD(J1,J2) - HTB
               DHT2(IE,KD) = HT2(IE,KD) - HT(IE)
               IF( DISTB(IE,KD) .EQ. 0.) GO TO 66
               SLOPE2(IE,KD) = DHT2(IE,KD) / DISTB(IE,KD)
                    IF( ANGLE .GT. 135 .AND. IH .EQ. IH1+200) THEN
                         NCA = 7
                    ELSE IF( ANGLE .LT. 45 .AND. IH .EQ. IH1-200) THEN
                         NCA = 7
```

```
                    ELSE IF( KD .GT. 1 ) THEN
                         JA = XB(IE,KD-1) / 100
                         JB = YB(IE,KD-1) / 100
                         NCA = NCODE( JA,JB)
                    END  IF
               NCB = NCODE(J1,J4)
66        IF ( CODE(NCA,NCB) .EQ. 1 .AND. SLOPE2(IE,KD) .LE. THRES1
     *         .AND. SLOPE2(IE,KD) .GE. THRES2 .AND. KD .LE. 20) THEN
                 IF( ANGLE .GT. 135.) THEN
                         IH = IH + 100
                 END  IF
                 IF( ANGLE .LT.  45.) THEN
                         IH = IH - 100
                 END  IF
                 IF( IH .LE. 0 ) GO TO 60
                 KD = KD + 1
                 GO TO 64
          ELSE IF (CODE(NCA,NCB) .NE. 1 .OR. SLOPE2(IE,KD) .GT. THRES1
     *         .OR. SLOPE2(IE,KD) .LT. THRES2 .OR. KD .GT. 20) THEN
                 IF(DISTB(IE,KD) .LT. RMIN) THEN
                         RMIN = DISTB(IE,KD)
                 END  IF
*    APPLY SPECIAL RULE(WIDTH DETERMINATION RULE 2) FOR BOUNDARY.
                 IF( NSPCL .EQ. 1 ) THEN
                         IF( KD .EQ. 2 ) THEN
                                 RMIN = 0.
                         END  IF
                 END  IF
                 KD = KD + 1
          END  IF
67               NO(IE) = KD - 1
60     CONTINUE
       END  IF
*      DETERMINE THE TOTAL ARC WIDTH.
*      ASSIGN THE WIDTH OF ROAD ITSELF.
               ROADW(1) = 18.
               ROADW(2) =  6.
               ROADW(3) =  4.
               ROADW(4) =  4.
               ROADW(5) =  2.
               ROADW(6) =  5.
               ROADW(7) =  4.
       IF (ARCTP .NE. 7) THEN
               WIDTH = LMIN + RMIN + ROADW(ARCTP)
       ELSE IF (ARCTP .EQ. 7) THEN
               WIDTH = ROADW(ARCTP)
       END  IF
       RETURN
       END
*************************************************************************

* I. SUBROUTINE FOR COMPUTING COORDINATES FOR THE MINIMUM DISTANCE.

*************************************************************************

       SUBROUTINE MIND(K,XM,YM,XA,YA,LMIN,RMIN,XL,YL,XR,YR)

         REAL       LMIN
         DIMENSION  XA(90),YA(90)
         DIMENSION  XL(10),YL(10),XR(10),YR(10)
                MK  = 1

*      ANGLE OF INCLINATION IS BETWEEN 0 AND 89.9999 DEGREES.
*      TAN(89.9999) = 572957.7
         IF ( XM .GE. 0. .AND. XM .LT. 572958.0) THEN
              DO  80 J = 1, K, K-1
                   INY = YA(J)
                   INX = XA(J)
                   YL(MK) = YA(J) + SQRT( LMIN**2 / ((1./YM **2)+1.))
                   INA = YL(MK)
```

```
                IF ( INA .EQ. 0 ) GO TO 80
                XL(MK) = XCORD( YM,INA,INY,INX )
                YR(MK) = YA(J) - SQRT( RMIN**2 / ((1./YM **2)+1.))
                INA = YR(MK)
                IF ( INA .EQ. 0 ) GO TO 80
                XR(MK) = XCORD( YM,INA,INY,INX )
                MK = MK + 1
80          CONTINUE
*     ANGLE OF INCLINATION IS BETWEEN 90.0001 AND 179.99 DEGREES.
      ELSE IF ( XM .LT. 0.) THEN
            DO  81 J = 1, K, K-1
                INY = YA(J)
                INX = XA(J)
                YL(MK) = YA(J) - SQRT( LMIN**2 / ((1./YM **2)+1.))
                INA = YL(MK)
                IF ( INA .EQ. 0 ) GO TO 81
                XL(MK) = XCORD( YM,INA,INY,INX )
                YR(MK) = YA(J) + SQRT( RMIN**2 / ((1./YM **2)+1.))
                INA = YR(MK)
                IF ( INA .EQ. 0 ) GO TO 81
                XR(MK) = XCORD( YM,INA,INY,INX )
                MK = MK + 1
81          CONTINUE
*     ANGLE OF INCLINATION IS 90 DEGREES.
       ELSE IF ( XM .GE. 572958.0) THEN
            DO  82 J = 1, K, K-1
                INY = YA(J)
                INX = XA(J)
                YL(MK) = YA(J)
                INA = YL(MK)
                IF ( INA .EQ. 0 ) GO TO 82
                XL(MK) = INX - LMIN
                YR(MK) = YA(J)
                INA = YR(MK)
                IF ( INA .EQ. 0 ) GO TO 82
                XR(MK) = INX + RMIN
                MK = MK + 1
82          CONTINUE
       END  IF
       RETURN
       END
*************************************************************************

* J. SUBROUTINE FOR COMPUTING THE RATE OF FLOW.

*************************************************************************

      SUBROUTINE FLOW(SP,WIDTH,ARCTP,CASE,FR,MAXNO)

       INTEGER    ARCTP,CASE
       DIMENSION  SP(7,3,2),FR(5,5),DSTN(6),MAXNO(3,3)
       DIMENSION  DW(3,2),DD(3,2)
       DATA       DSTN   / 25.,5.,20.,5.,5.,3. /
*   DSTN IS DISTANCE BETWEEN EACH ELEMENTS.

            I  = ARCTP
*   ASSIGN DOCTRINAL WIDTH AND DEPTH OF BATTALION.
            DW(1,1) = 1.
            DD(1,1) = 3.
            DW(1,2) = 0.005
            DD(1,2) = 4.
            DW(2,1) = 1.
            DD(2,1) = 3.
            DW(2,2) = 0.005
            DD(2,2) = 4.
            DW(3,1) = 1.
            DD(3,1) = 3.
            DW(3,2) = 0.006
            DD(3,2) = 2.5
            WIDTH1  = WIDTH / 1000.
```

```
*     COMPUTE THE FLOW RATE.( BATTALION / HOUR )
      IF ( CASE .EQ. 1 ) THEN
            DO  80    J = 1,2
                      K = 1
                      D = DW(J,K) * DD(J,K)
                      FR(J,K) = ( SP(I,J,K) * WIDTH1 ) / D
80          CONTINUE
            DO  81    J = 1,2
                      K = 2
                IF(WIDTH  .GE. 5.) THEN
                      WIDTH1 = 0.005
                END   IF
                      D = DW(J,K) * DD(J,K)
                      FR(J,K) = ( SP(I,J,K) * WIDTH1 ) / D
81          CONTINUE
      ELSE IF ( CASE .EQ. 2 ) THEN
            J = 3
            DO  83  K = 1,2
                IF(K .EQ. 2 .AND. WIDTH1 .GE. 0.006) THEN
                      WIDTH1 = 0.006
                ELSE  IF(K .EQ. 2 .AND. WIDTH1 .LT. 0.006) THEN
                      WIDTH1 = WIDTH1
                END   IF
                D = DW(J,K) * DD(J,K)
                FR(J,K) = ( SP(I,J,K) * WIDTH1 ) / D
83          CONTINUE
      END IF

*     COMPUTE THE MAX NUMBER OF ELEMENT IN MULTIPLE COLUMN FORMATION.
        IF ( CASE .EQ. 1 ) THEN
              KNA = 1
              DO  85  KA = 1,2
              DO  85  KB = 1,2
                  IF( KB .EQ. 1) THEN
                        MAXNO(KA,KB) = WIDTH / DSTN(KNA)
                  ELSE IF(KB .EQ. 2) THEN
                        MAXNO(KA,KB) = (WIDTH1 * 1100.) / DSTN(KNA)
                  END IF
                  KNA = KNA + 1
85            CONTINUE
          ELSE IF ( CASE .EQ. 2 ) THEN
              KNA = 5
              KA  = 3
              DO  86  KB = 1,2
                  IF( KB .EQ. 1) THEN
                        MAXNO(KA,KB) = WIDTH / DSTN(KNA)
                  ELSE IF(KB .EQ. 2) THEN
                        MAXNO(KA,KB) = (WIDTH1 * 1000) / DSTN(KNA)
                  END IF
                  KNA = KNA + 1
86            CONTINUE
          END IF
      RETURN
      END
*******************************************************************************

* K. SUBROUTINE FOR COMPUTING THE MIN TRAVERSAL TIME ALONG THE ARC.

*******************************************************************************

      SUBROUTINE TIME(ARCTP,SP,UNITF,CASE,DIST,WIDTH,ATIME)

      INTEGER    ARCTP,CASE,UNITF
      REAL       KR
      DIMENSION  ATIME(3),SP(7,3,2)
      DATA       KR / 0.9 /
*  KR IS COEFFICIENT OF ROUTE AVAILABILITY.

      I     = ARCTP
```

```
      ADIST = DIST  / 1000.
      IF ( UNITF .EQ. 1 ) THEN
           IF( CASE .EQ. 1 ) THEN
               DO  90  KU = 1,2
                       ATIME(KU) = ADIST /( KR * SP(I,KU,1))
90             CONTINUE
           ELSE IF( CASE .EQ. 2 ) THEN
                       ATIME(3) = ADIST / SP(I,3,1)
           END  IF
      ELSE IF ( UNITF .EQ. 2 ) THEN
           IF( CASE .EQ. 1 ) THEN
               K = 2
               DO  95  J = 1,2
                       ATIME(J) = ADIST / SP(I,J,K)
95             CONTINUE
           ELSE IF( CASE .EQ. 2 ) THEN
                       ATIME(3) = ADIST / SP(I,3,2)
           END IF
      END  IF
*     IF ARC WIDTH IS LESS THAN MIN FORMATION WIDTH,
*     MAKE THE TIME BIG NUMBER.
      BIG   = 9.99
      FORWV = 4.
      FORWD = 2.
      IF ( UNITF .EQ. 1 ) THEN
           IF( CASE .EQ. 1 ) THEN
               IF( WIDTH .LT. FORWV ) THEN
                   DO  97  KN = 1,2
                       ATIME(KN) = BIG
97                 CONTINUE
               END  IF
           ELSE IF( CASE .EQ. 2 ) THEN
               IF( WIDTH .LT. FORWD ) THEN
                   ATIME(3) = BIG
               END IF
           END IF
      END  IF
      RETURN
      END
*************************************************************************

* L. SUBROUTINE FOR PRINTING THE RESULTS.

*************************************************************************

      SUBROUTINE PRT

      INTEGER    ALTD,ARCTP,CASE,CODE,PRINT,UTMX,UTMY,UNITF,SPD
      REAL       LMIN
      DIMENSION  ALTD(100,100),NCODE(100,100),NX(10),NCODEX(10)
      DIMENSION  XNODE(90),YNODE(90),CODE(0:7,0:7)
      DIMENSION  DISTA(90),XA(90),YA(90)
      DIMENSION  HT(90),DHT(90),SLOPE1(90)
      DIMENSION  DISTB(100,100),XB(100,100),YB(100,100)
      DIMENSION  HT2(100,100),DHT2(100,100),SLOPE2(100,100)
      DIMENSION  NO(90),SLPCH(90)
      DIMENSION  XL(10),YL(10),XR(10),YR(10)
      DIMENSION  SP(7,3,2),FR(5,5),MAXNO(3,3)
      DIMENSION  ACTD(90),ATIME(3)
      DIMENSION  ROADW(7)
      COMMON / INTS  / NSTEP,IX1,IY1,IX2,IY2,JA,JB,K,NARC,NCOUNT,
     *                 NODEA,NODEB,J1,J2,J3,J4,PRINT,UTMX,UTMY,
     *                 CASE,ARCTP,FR,AD,UNITF,NSPCL
      COMMON / INTA  / NCODE,NO,ALTD,CODE,MAXNO
      COMMON / SREAL / XM,ANGLE,DIST,TSLPCH,ASLPCH,ALT1,ALT2,AVGSLP,
     *                 LMIN,RMIN,XL,YL,XR,YR,YM,RY,WIDTH,THRES1,THRES2
      COMMON / AREAL / XA,YA,HT,DHT,DISTA,DISTB,SLPCH,SLOPE1,SLOPE2,
     *                 XB,YB,HT2,DHT2,SP,ACTD,ATIME,ROADW
       OPEN ( UNIT = 10, FILE = 'NETW1' )
       OPEN ( UNIT = 20, FILE = 'NETW2' )
```

```
      OPEN ( UNIT = 25, FILE = 'NETA'  )
      OPEN ( UNIT = 26, FILE = 'NETB'  )

*   PRINT  GENERAL INFORMATION FOR EACH ARC.
      IF ( PRINT .EQ. 1 ) THEN
      WRITE(6,700) NCOUNT,NODEA,NODEB
700   FORMAT(' ',/,7X,' ARC     : ',I2,
     * //,4X,71('-'),//,5X,'NODE  ',I2,6X,'NODE  ',I2,
     * 5X,' XM      ANGLE      DISTANCE     ACT. DISTANCE',
     *  /,4X,' IX1   IY1',5X,'IX2   IY2')
      WRITE(6,701) IX1,IY1,IX2,IY2,XM,ANGLE,DIST,AD
701   FORMAT(' ',1X,2I6,2X,2I6,1X,F8.2,F8.2,2X,F11.2,2X,F10.2)
      WRITE(6,730)
730   FORMAT(' ',/,4X,71('-'),//,20X,'SOME  POINTS  ALONG  ARC',/,
     * 4X,'POSITION ',7X,'COORDINATE( X  :  Y )',5X,'ALTITUDE',/)
      DO  91  MA = 1,K
          WRITE(6,731) MA,XA(MA),YA(MA),HT(MA)
731       FORMAT(' ',3X,'POINT ',I3,8X,'(',2F8.1,1X,')', F13.1)
91    CONTINUE
      WRITE(6,732)
732   FORMAT(' ',/,4X,71('-'),/,
     * 7X,'POSITION ',8X,'DELTA H',5X, 'DISTANCE',6X,'SLOPE',5X,
     * 'ACTUAL DISTANCE')
      DO  92  MB = 1,K-1
          WRITE(6,733) MB,MB+1,DHT(MB),DISTA(MB),SLOPE1(MB),ACTD(MB)
733       FORMAT(' ',3X,'POINT ',I3,' -',I3, 2F12.1,F12.3,F12.1)
92    CONTINUE
      WRITE(6,734)
734   FORMAT(' ',/,4X,71('-'),/,
     * 7X,'POSITION ',5X,' SLOPE   CHANGE')
      DO  93  MC = 1,K-2
          WRITE(6,735) MC,MC+1,SLPCH(MC)
735       FORMAT(' ',3X,'POINT ',I3,' -',I3,2X, F11.2)
93    CONTINUE
      WRITE(6,738) TSLPCH,ASLPCH
738   FORMAT(' ', /,4X,'TOTAL SLOPE CHANGE :  ',F10.4,
     * /,4X,'AVERAGE SLOPE CHG. :  ',F10.4,/,4X,71('-'))
      WRITE(6,740) NODEA,NODEB
740   FORMAT(' ',/,4X,'NODE',I3,':    ALTITUDE 1  ',
     *                4X,'NODE',I3,':    ALTITUDE 2  ')
      WRITE(6,741) ALT1,ALT2
741   FORMAT(' ',20X,F5.1,23X,F5.1)
      WRITE(6,742) AVGSLP
742   FORMAT(' ',/,4X,'AVERAGE SLOPE    :   ',F7.4,/,4X,71('-'),/ )
      WRITE(6,750)
750   FORMAT(' ',15X,' SOME  POINTS  OFF  ARC ( FROM EACH POINT ',
     * 'ON THE ARC )',/,7X,'COORDINATE ( X  :  Y )',8X,'ALTITUDE',
     * 4X,'DELTA H',4X,'DISTANCE',5X,'SLOPE',/)

            N1 = 0
      DO  94  JX = 1,K
            N1 = N1+ 1
      DO  94  JY = 1,NO(N1)
          WRITE(6,751) JX,JY,XB(JX,JY),YB(JX,JY),HT2(JX,JY),
     *    DHT2(JX,JY),DISTB(JX,JY),SLOPE2(JX,JY)
751       FORMAT(' ',3X,'POINT ',I2,' -',I2,' (',2F7.1,1X,')',4F11.2)
94    CONTINUE
      WRITE(6,752) LMIN,RMIN,WIDTH
752   FORMAT(' ',/,4X,71('-'),/,25X,'MINIMUM  DISTANCE  OFF  ARC',
     * /,4X,'LEFT  MIN DISTANCE   :',F8.1,
     * /,4X,'RIGHT MIN DISTANCE   :',F8.1,
     * /,4X,'ARC   WIDTH          :',F8.1)
      WRITE(6,753)
753   FORMAT(' ',/,27X,'POINT 1',13X,'POINT 2',
     *  /,24X,'X LEFT    Y LEFT',4X,'X RIGHT   Y RIGHT')
      WRITE(6,754)(XL(K),YL(K),XR(K),YR(K), K=1,2)
754   FORMAT(' ',/,4X,'INITIAL POINT : ',4F10.1,
     *          /,4X,'END     POINT : ',4F10.1)
      WRITE(6,756)
756   FORMAT(' ',/,4X,71('*'))
```

```
*   PRINT FLOW RATE AND TRAVERSAL TIME.
       ELSE IF ( PRINT .EQ. 2 ) THEN
       WRITE(20,760) NCOUNT,NODEA,NODEB
760    FORMAT(' ',/,7X,' ARC      : ',I2,//,
     * 4X,71('-'),//,5X,'NODE  ',I2,6X,'NODE  ',I2,
     * 6X,'      XM        ANGLE      DISTANCE',
     *  /,4X,' IX1   IY1',5X,'IX2   IY2')
       WRITE(20,761) IX1,IY1,IX2,IY2,XM,ANGLE,DIST
761    FORMAT(' ',1X,2I6,2X,2I6,3X,3F11.2)
       WRITE(20,762) LMIN,RMIN,ROADW(ARCTP),WIDTH,ARCTP
762    FORMAT(' ',/,4X,71('-'),/,25X,'MINIMUM  DISTANCE  OFF  ARC',
     * /,4X,'LEFT  MIN DISTANCE    :',F8.1,
     * /,4X,'RIGHT MIN DISTANCE    :',F8.1,
     * /,4X,'WIDTH OF ROAD ITSELF :',F8.1,
     * //,4X,'ARC   WIDTH           :',F8.1,
     * /,4X,'ARC   TYPE            :',I8,
     * /,4X,'UNIT TYPE',4X,'MISSION TYPE',3X,
     * 'FLOW RATE',3X,'MAX # ELEMENT')
           IF ( CASE .EQ. 1) THEN
              DO  95  IJ = 1,2
              DO  95  IK = 1,2
                  WRITE(20,766) IJ,IK,FR(IJ,IK),MAXNO(IJ,IK)
766               FORMAT(' ',7X,I2,9X,I2,10X,F7.2,9X,I4)
95            CONTINUE
           ELSE IF ( CASE .EQ. 2) THEN
                     IJ = 3
              DO  96  IK = 1,2
                  WRITE(20,768) IJ,IK,FR(IJ,IK),MAXNO(IJ,IK)
768               FORMAT(' ',7X,I2,9X,I2,10X,F6.1,9X,I4)
96            CONTINUE
           END IF
        WRITE(20,7620)
7620    FORMAT(' ',/,4X,'UNIT TYPE   MIN TRAVERSAL TIME')
           IF ( CASE .EQ. 1) THEN
              DO  950  K = 1,2
                  WRITE(20,7660) K,ATIME(K)
7660               FORMAT(' ',7X,I2,9X,F9.2)
950            CONTINUE
           ELSE IF ( CASE .EQ. 2) THEN
                   IK = 3
                   WRITE(20,7680) IK,ATIME(IK)
7680               FORMAT(' ',7X,I2,9X,F9.2)
           END IF
       WRITE(20,763)
763    FORMAT(' ',/,27X,'POINT 1',13X,'POINT 2',
     *  /,24X,'X LEFT    Y LEFT',4X,'X RIGHT   Y RIGHT')
       WRITE(20,764) XL(1),YL(1),XR(1),YR(1)
764    FORMAT(' ',/,4X,'INITIAL POINT : ',4F10.1)
       WRITE(20,765) XL(2),YL(2),XR(2),YR(2)
765    FORMAT(' ',3X,'END     POINT : ',4F10.1)
       WRITE(20,767)
767    FORMAT(' ',/,4X,71('*'))

*   PRINT COORDINATES OF BOUNDARY LINE.
       ELSE IF ( PRINT .EQ. 4 ) THEN
            DO 97  IN = 1,2
               XL(IN) = UTMX +(XL(IN) /1000.)
               YL(IN) = UTMY +(YL(IN) /1000.)
               XR(IN) = UTMX +(XR(IN) /1000.)
               YR(IN) = UTMY +(YR(IN) /1000.)
97          CONTINUE
       WRITE(10,774) NODEA, XL(1),YL(1),XR(1),YR(1)
774    FORMAT(' ',/,4X,'NODE',I2,'    ',4F10.2)
       WRITE(10,775) NODEB, XL(2),YL(2),XR(2),YR(2)
775    FORMAT(' ',3X,'NODE',I2,'    ',4F10.2)

*   PRINT NETWORK DATA FOR CARTESIAN SPACE NETWORK PROGRAM.
       ELSE IF ( PRINT .EQ. 5 ) THEN
            IF( NODEA .GT. NODEB ) THEN
```

```
                NODE1 = NODEB
                NODE2 = NODEA
            ELSE IF( NODEB .GT. NODEA ) THEN
                NODE1 = NODEA
                NODE2 = NODEB
            END IF
                DST = DIST  / 1000.
                WD  = WIDTH / 1000.

                   IF( CASE .EQ. 1  ) THEN
                        KS = 1
                        KT = 1
                   ELSE IF( CASE .EQ. 2  ) THEN
                        KS = 3
                        KT = 1
                   END IF
            NTIME =  ATIME(KS) * 100
            NFR   =  FR(KS,KT) * 100
                  IARCT  = ARCTP
                  IUNITF = UNITF
            SPD = SP( IARCT, KS, IUNITF )
            IF( CASE .EQ. 1  ) THEN
          WRITE(25,874) NCOUNT,NODE1,NODE2,NTIME,NFR,DST,WD,SPD
            ELSE IF( CASE .EQ. 2  ) THEN
          WRITE(26,874) NCOUNT,NODE1,NODE2,NTIME,NFR,DST,WD,SPD
            END  IF
874       FORMAT(' ',1X,I3,2I5,2I6,3X,F4.1,3X,F6.3,2X,I4)
          END  IF
          RETURN
          END
```

APPENDIX D
COMPUTER PROGRAM FOR CARTESIAN SPACE NETWORK

This appendix contains the computer program used to support the algorithms in Chapter IV.

```
      PROGRAM  NET
*     JULY    13,  1987  (13:00)
*     NETWORK MAIN PROGRAM
*     SHORTEST  PATH  ALGORITHM

*     *************  VARIABLE DECLARATION  *************

      INTEGER    TAIL,HEAD,TIME,FLOW
      INTEGER    CASE,NTYPE,PARAM,SP,UNITF,ROWSP
      CHARACTER  *1 RESM
      DIMENSION  TAIL(150),HEAD(150),TIME(150)
      DIMENSION  FLOW(100),DIST(100),WIDTH(100)
      DIMENSION  LINK(100,100),LARC(100)
      DIMENSION  NO(100),SP(100)
      DIMENSION  INODE(150),JNODE(150),ARCOST(150)
      DIMENSION  ISPATH(50)
         COMMON / INTS / CASE,NTYPE,N,M,NARC,ISTART,LAST,PARAM,
     *                   NUMP,UNITF,NVEH,NCOL,NROW,ROWSP,MINE,
     *                   NWD,NDP,NFORWD
         COMMON / INTA / TAIL,HEAD,TIME,FLOW,LINK,LARC,NO,SP,
     *                   INODE,JNODE,ISPATH
         COMMON / REALS/ BIG,TPATH,XLEN,PATHL,FORWD,FORDP,PRT
         COMMON / REALA/ WIDTH,DIST,RESM
*-------------------------------------------------------------------
* (1). SET THE INITIAL OPTIONS.
*-------------------------------------------------------------------

      CALL OPTION (CASE,UNITF,NCOL,ROWSP,PARAM,FORWD,NFORWD,
     *             RESM,MINE)
*                    SET THE START NODE
           ISTART =  1
*                    SET THE LAST NODE
           LAST   = 34
*                    TOTAL NUMBER OF ARC IN THE SECTOR
           NARC   = 72
                    TOTAL NUMBER OF NODE IN THE SECTOR
           NNODE = 34
*    SET THE INITIAL CONDITIONS.
      CALL INIT (ARCOST,NUMP,ISPATH,XLEN,ADJUST,PATHL,SPEED,
     *           LINK,TPATH,NNODE,BIG)
*-------------------------------------------------------------------
* (2). READ THE DATA FROM THE DATA FILE.(DATA 1, OR  DATA 2 )
*-------------------------------------------------------------------

*          DATA 1 ; ARCS AND CHARACTERISTICS FOR VEHICLE UNIT
*          DATA 2 ; ARCS AND CHARACTERISTICS FOR DISMOUNTED TROOPS

*   DETERMINE VARIABLE FOR READING INPUT DATA.( VEHICLE OR DISMOUNTED )
           IF( PARAM .EQ. 1 ) THEN
                IF( CASE .EQ. 1 ) THEN
                     KUNIT = 1
                ELSE IF( CASE .EQ. 2 ) THEN
                     KUNIT = 2
                END IF
```

```
               ELSE IF( PARAM .EQ. 2 ) THEN
                    KUNIT = 1
               END  IF
          READ(KUNIT,100, END=999) (NO(I),TAIL(I),HEAD(I),TIME(I),
     *                         FLOW(I),DIST(I),WIDTH(I),SP(I),I=1,NARC)
100       FORMAT(2X,I3,2I5,2I6,3X,F4.1,3X,F6.3,2X,I4)

*     SPECIFY THE WIDTH AND DEPTH OF UNIT FORMATION.
999       CALL  FORMTN (NARC,CASE,UNITF,NCOL,ROWSP,PARAM,WIDTH,TIME,
     *                  NROW,DIST,FLOW,FORWD,FORDP,NVEH,NWD,NDP)
*-------------------------------------------------------------------

* (3). DETERMINE THE APPROPRIATE VALUE FOR PARAMETER.
*-------------------------------------------------------------------

*     DETERMINE THE VALUE OF PARAMETER WHEN THE MINEFIELD IS ON ARC.
               IF (RESM .EQ. 'Y') THEN
                    IF (PARAM .EQ. 1) THEN
                         TIME(MINE) = TIME(MINE)* 4
                    END  IF
               END  IF
*      DETERMINE THE APPROPRIATE VALUE FOR PARAMETER
          N = NNODE
          M = NARC
          DO  10  KA = 1,NARC
               INODE(KA) = TAIL(KA)
               JNODE(KA) = HEAD(KA)
               IF (PARAM .EQ. 1) THEN
                    ARCOST(KA) = TIME(KA)
               ELSE IF (PARAM .EQ. 2) THEN
                    ARCOST(KA) = DIST(KA)
               END  IF
10        CONTINUE
*-------------------------------------------------------------------

* (4). DETERMINE THE MIN (PARAMETER) PATH.
*-------------------------------------------------------------------

     CALL  SHORTP (N,M,INODE,JNODE,ARCOST,ISTART,LAST,BIG,NUMP,
     *                   ISPATH,XLEN,NP )

*  DETERMINE THE ARC NUMBER ALONG THE PATH.
               DO   60  M3 = 1, NARC
                    LINK(TAIL(M3),HEAD(M3)) = NO(M3)
60             CONTINUE
               PATHL = 0.
          DO   70  MA = 1, NUMP-1
               LARC(MA) = LINK(ISPATH(MA), ISPATH(MA+1))
               PATHL = PATHL + DIST( LARC(MA) )
70        CONTINUE

*  WHEN 'PRT' IS 1.0, THERE IS NO FEASIBLE ROUTE TO GO.
          IF (PARAM .EQ. 1) THEN
               PRT = 0.0
               DO 75 MB = 1,NUMP-1
                    IF (TIME(LARC(MB)) .GE. 999) THEN
                         PRT = 1.0
                    END  IF
75             CONTINUE
          END  IF

*  DETERMINE THE TOTAL TRAVERSAL TIME FROM START NODE TO LAST NODE.
                    SPEED = 0.
               DO   80  M5 = 1, NUMP-1
                    SPEED = SPEED + (DIST(M5) * SP(M5) / PATHL )
80             CONTINUE
               ADJUST =  FORDP  / SPEED
               TPATH  =  XLEN  + ( ADJUST * 100. )
*-------------------------------------------------------------------
```

```
* (5).  PRINT THE RESULT OF CALCULATION.
*-----------------------------------------------------------------------

         CALL PRNT
              STOP
              END
**********************************************************************

* A.  SUBROUTINE FOR DETERMINING THE INITIAL OPTIONS.

**********************************************************************

       SUBROUTINE  OPTION (CASE,UNITF,NCOL,ROWSP,PARAM,FORWD,NFORWD,
     *                     RESM, MINE)
         INTEGER    CASE,PARAM,UNITF,ROWSP
         CHARACTER  *1 RESM

           WRITE(6,603)
603        FORMAT(' ',///)
15         PRINT*, '
           PRINT*, '          WHICH TYPE OF PARAMETER DO YOU DESIRE?'
           PRINT*, '
           PRINT*, '          *********************************'
           PRINT*, '          *                                * '
           PRINT*, '          *   1. MIN TIME PATH             *'
           PRINT*, '          *                                * '
           PRINT*, '          *   2. MIN DISTANCE PATH         *'
           PRINT*, '          *                                *'
           PRINT*, '          *********************************'
           READ(5,*)  PARAM
           PRINT*, '** NOTE : FOR YOUR REFERENCE, CURRENT ANSWER IS',
     *          PARAM
                IF ( PARAM .EQ. 1 ) THEN
                   GO TO 16
                ELSE IF ( PARAM .EQ. 2 ) THEN
                   GO TO 20
                ELSE IF ( PARAM .NE. 1 .OR. PARAM .NE. 2 ) THEN
                   PRINT*, '
                   PRINT*, '*** ERROR : ENTER THE NUMBER 1 OR 2 ***'
                   PRINT*, '
                   GO TO  15
                END IF
16         WRITE(6,601)
601        FORMAT(' ',////////)
           PRINT*, '          WHICH TYPE OF UNIT DO YOU DESIRE? ( 1 OR 2 )'
           PRINT*, '
           PRINT*, '          *********************************'
           PRINT*, '          *                                * '
           PRINT*, '          *   1. VEHICLE UNIT              *'
           PRINT*, '          *                                * '
           PRINT*, '          *   2. DISMOUNTED TROOPS         *'
           PRINT*, '          *                                * '
           PRINT*, '          *********************************'
           READ(5,*)  CASE
           PRINT*, '** NOTE : FOR YOUR REFERENCE, CURRENT ANSWER IS',
     *          CASE
                IF ( CASE .EQ. 1 .OR. CASE .EQ. 2 ) THEN
                GO TO 17
                ELSE IF ( CASE .NE. 1 .OR. CASE .NE. 2 ) THEN
           PRINT*, '
           PRINT*, '*** ERROR : ENTER THE NUMBER 1 OR 2 ***'
           PRINT*, '
                GO TO  16
                END IF
17         WRITE(6,602)
602        FORMAT(' ',////////)
           PRINT*, '          WHICH TYPE OF FORMATION DO YOU DESIRE? '
           PRINT*, '
           PRINT*, '          ************************************'
           PRINT*, '          *                                  * '
```

```
      PRINT*, '         *   1. MULTIPLE COLUMN FORMATION   *'
      PRINT*, '         *                                  * '
      PRINT*, '         *   2. SINGLE COLUMN  FORMATION    *'
      PRINT*, '         *                                  * '
      PRINT*, '         ************************************'
      READ(5,*)  UNITF
      PRINT*, '** NOTE : FOR YOUR REFERENCE, CURRENT ANSWER IS',
     *          UNITF
           IF ( UNITF .EQ. 1 .OR. UNITF .EQ. 2 ) THEN
           GO TO 18
           ELSE IF ( UNITF .NE. 1 .OR. UNITF .NE. 2 ) THEN
      PRINT*, '                '
      PRINT*, '*** ERROR : ENTER THE NUMBER 1 OR 2 ***'
      PRINT*, '                '
           GO TO  17
           END IF
18    IF (CASE .EQ. 1) THEN
           IF (UNITF .EQ. 1) THEN
      WRITE(6,604)
604   FORMAT(' ',/////////)
      PRINT*, '         ENTER THE NUMBER OF COLUMN.(5, 10, 20)'
      PRINT*, '  '
      PRINT*, '             *   *   *   *   *   *   *'
      PRINT*, '                                     '
      PRINT*, '             *   *   *   *   *   *   *'
      PRINT*, '                                     '
      PRINT*, '             *   *   *   *   *   *   *'
      READ(5,*) NCOL
      PRINT*, '  '
      WRITE(6,605)
605   FORMAT(' ',/////////)
      PRINT*, '         ENTER THE DISTANCE BETWEEN EACH ROW.(75,50,25)'
      PRINT*, '  '
      PRINT*, '        /    *   *   *   *   *   *   *'
      PRINT*, '        ||                           '
      PRINT*, '         /   *   *   *   *   *   *   *'
      PRINT*, '                                      '
      PRINT*, '             *   *   *   *   *   *   *'
      PRINT*, ' '
      READ(5,*) ROWSP
           ELSE IF (UNITF .EQ. 2) THEN
      WRITE(6,606)
606   FORMAT(' ',/////////)
      PRINT*, '         ENTER THE DISTANCE BETWEEN EACH ROW.(75,50,25)'
      PRINT*, '  '
      PRINT*, '        /    *   *   *   *   *   *   *'
      PRINT*, '        ||                           '
      PRINT*, '         /   *   *   *   *   *   *   *'
      PRINT*, '                                      '
      PRINT*, '             *   *   *   *   *   *   *'
      PRINT*, ' '
      READ(5,*) ROWSP
           END IF
      ELSE IF (CASE .EQ. 2) THEN
           IF (UNITF .EQ. 1) THEN
      WRITE(6,614)
614   FORMAT(' ',/////////)
      PRINT*, '         ENTER THE NUMBER OF COLUMN.(10, 20, 50)'
      PRINT*, '  '
      PRINT*, '             *   *   *   *   *   *   *'
      PRINT*, '                                     '
      PRINT*, '             *   *   *   *   *   *   *'
      PRINT*, '                                     '
      PRINT*, '             *   *   *   *   *   *   *'
      READ(5,*) NCOL
      PRINT*, '  '
      WRITE(6,615)
615   FORMAT(' ',/////////)
      PRINT*, '         ENTER THE DISTANCE BETWEEN EACH ROW.(5,10,15)'
      PRINT*, '  '
```

```
          PRINT*, '        /   *   *   *   *   *   *   *'
          PRINT*, '       ||                           '
          PRINT*, '        /  *   *   *   *   *   *   *'
          PRINT*, '                                     '
          PRINT*, '           *   *   *   *   *   *   *'
          PRINT*, ' '
          READ(5,*) ROWSP
               END IF
       END IF
20        WRITE(6,607)
607       FORMAT(' ',/////)
          PRINT*, '          IS THERE ANY MINEFIELD ALONG THE ARC? (Y OR N)'
          READ(5,'(A1)')  RESM
               IF (RESM .EQ. 'Y') THEN
          PRINT*, '          ENTER THE ARC NUMBER OF MINEFIELD ARC'
          READ(5,*)  MINE
               ELSE IF (RESM .EQ. 'N') THEN
                    MINE = 0
               END IF
          RETURN
          END
*************************************************************************

* B.  SUBROUTINE FOR DETERMINING THE INITIAL CONDITIONS.

*************************************************************************

       SUBROUTINE INIT(ARCOST,NUMP,ISPATH,XLEN,ADJUST,PATHL,SPEED,
      *                LINK,TPATH,NNODE,BIG)

         DIMENSION  ISPATH(50),ARCOST(100)
         DIMENSION  LINK(100,100)

           BIG    = 1.0E10
           NUMP = 0
           XLEN = 0.
           ADJUST = 0.
           PATHL  = 0.
           SPEED  = 0.
           TPATH  = 0.
           DO  10 I = 1,NNODE
                  ISPATH(I) = 0
10         CONTINUE
           DO  20 J = 1,100
                  ARCOST(J) = 0.
20         CONTINUE
           DO   50  M1 = 1, 100
           DO   50  M2 = 1, 100
                LINK(M1,M2) = 0
50         CONTINUE
           RETURN
           END
*************************************************************************

* C.  SUBROUTINE FOR DETERMINING THE FORMATION WIDTH AND DEPTH.

*************************************************************************

       SUBROUTINE  FORMTN (NARC,CASE,UNITF,NCOL,ROWSP,PARAM,WIDTH,TIME,
      *                    NROW,DIST,FLOW,FORWD,FORDP,NVEH,NWD,NDP)

         INTEGER    TIME,FLOW
         INTEGER    CASE,PARAM,UNITF,ROWSP
         DIMENSION  TIME(100),FLOW(100),DIST(100),WIDTH(100)
         BIG1 = 999.

*_______________________________________________________________________

* (A).  DETERMINE THE FORMATION WIDTH OF A UNIT OF MOVEMENT.
*_______________________________________________________________________
```

```
      IF( CASE .EQ. 1 ) THEN
            NVEH = 50
            IF ( UNITF .EQ. 1 ) THEN
                 NROW = NVEH / NCOL
                 COLSP = 25.0
                 FORWD = ((NCOL-1) * COLSP) / 1000
            ELSE IF( UNITF .EQ. 2 ) THEN
                 NROW = 50
                 NCOL = NVEH / NROW
                 COLSP = 4.0
                 FORWD = (NCOL * COLSP) / 1000
            END  IF
                 FORDP = ((NROW-1) * ROWSP) / 1000
      ELSE IF( CASE .EQ. 2 ) THEN
*        WHEN UNIT IS MULTIPLE  COLUMN FORMATION, THE FORMATION WIDTH
*        WAS ENTERED.
            IF ( UNITF .EQ. 1 ) THEN
                 NMEN = 1000
                 NROW = NMEN / NCOL
                 COLSP = 5.0
                 FORWD = (NCOL * COLSP) / 1000.
                 FORDP = (NROW * ROWSP) / 1000.
            ELSE IF ( UNITF .EQ. 2 ) THEN
                 FORWD = 0.006
                 FORDP = 2.5
            END IF
                 NWD = FORWD * 1000
                 NDP = FORDP * 1000
      END IF
*-----------------------------------------------------------------------
* (B).  ASSIGN BIG NUMBER WHEN ARC WIDTH IS LESS THAN FORMATION WIDTH.
*-----------------------------------------------------------------------
      DO   50   I = 1, NARC
            IF (PARAM .EQ. 1) THEN
                 IF (WIDTH(I) .LT. FORWD) THEN
                          TIME(I) = BIG1
                 END  IF
            END  IF
50    CONTINUE
      RETURN
      END

**************************************************************************
* D.  SUBROUTINE FOR DETERMINING THE SHORTEST PARAMETER PATH.
**************************************************************************

      SUBROUTINE SHORTP (N,M,INODE,JNODE,ARCOST,ISTART,LAST,BIG,NUMP,
     *                   ISPATH,XLEN,NP )

*    MINIMUM  PATH
*    FIND A SHORTEST PATH BETWEEN TWO GIVEN NODES

      INTEGER    INODE(M),JNODE(M), ISPATH(N)
      REAL       WK4,ARCOST(M)
      LOGICAL    IWORK1(34),IFIN
      DIMENSION  IWORK2(34),IWORK3(34),WK4(34)
*     34 :       NUMBER OF NODES IN NETWORK.

        DO   10   I = 1,N
             WK4(I) = BIG
             IWORK1(I) = .TRUE.
             IWORK2(I) = 0
10      CONTINUE
             WK4(ISTART) = 0.
             I  = ISTART
```

```
              IWORK1(ISTART) = .FALSE.
              NP = 0
              XLEN = 0.
*-----------------------------------------------------------------------
* (A) FOR EACH FORWARD ARC ORIGINATING AT NODE I CALCULATE THE LENGTH
*     OF THE PATH TO NODE I.
*-----------------------------------------------------------------------
20            IC = 0
              DO   30   K = 1,M
                   IF (INODE(K) .EQ. I) THEN
                       IC  = IC + 1
                       IWORK3(IC) = K
                       ISPATH(IC) = JNODE(K)
                   END IF
                   IF (JNODE(K) .EQ. I) THEN
                      IC  = IC + 1
                      IWORK3(IC) = K
                      ISPATH(IC) = INODE(K)
                   END IF
30            CONTINUE
              IF (IC .GT. 0 ) THEN
                   DO  40  L = 1,IC
                       K = IWORK3(L)
                       J = ISPATH(L)

                       IF (IWORK1(J)) THEN
                       D  = WK4(I) + ARCOST(K)
                             IF (D .LT. WK4(J)) THEN

                                  WK4(J) = D
                                  IWORK2(J) = K
                             END IF
                       END IF
40                 CONTINUE
              END  IF
*-----------------------------------------------------------------------
* (B).  FIND THE MINIMUM POTENTIAL.
*-----------------------------------------------------------------------

              D = BIG
              IENT = 0
              IFIN = .FALSE.
              DO   50   I = 1,   N
                   IF (IWORK1(I)) THEN
                        IFIN = .TRUE.
                        IF (WK4(I) .LT. D) THEN
                            D = WK4(I)
                            IENT  = I
                        END IF
                   END IF
50            CONTINUE
*-----------------------------------------------------------------------
* (C). INCLUDE THE NODE IN THE CURRENT PATH.
*-----------------------------------------------------------------------

              IF (D .LT. BIG) THEN
                   IWORK1(IENT)=.FALSE.
                   IF (IENT .NE. LAST) THEN
                        I = IENT
                        GO TO 20
                   END IF
              ELSE
                   IF (IFIN) THEN
                        NP = 1
                        RETURN
                   END IF
```

```
          END IF
               IJ = LAST
               NUMP = 1
               ISPATH(1) = LAST
60             K = IWORK2(IJ)
                    IF (INODE(K) .EQ. IJ) THEN
                         IJ= JNODE(K)
                    ELSE
                         IJ= INODE(K)
                    END IF
                    NUMP = NUMP + 1
                    ISPATH(NUMP) = IJ
               IF (IJ .NE. ISTART) GO TO 60
               L = NUMP / 2
               J = NUMP
          DO   70   I = 1,L
               K = ISPATH(I)
               ISPATH(I) = ISPATH(J)
               ISPATH(J) = K
70             J = J - 1
          XLEN = WK4(LAST)
          RETURN
          END

*************************************************************************

* E.  SUBROUTINE FOR PRINTING THE RESULTS OF CALCULATION.

*************************************************************************

      SUBROUTINE PRNT

       INTEGER    TAIL,HEAD,TIME,FLOW
       INTEGER    CASE,NTYPE,PARAM,SP,UNITF,ROWSP
       CHARACTER  *1 RESM
       DIMENSION  TAIL(150),HEAD(150),TIME(150)
       DIMENSION  FLOW(100),DIST(100),WIDTH(100)
       DIMENSION  LINK(100,100),LARC(100)
       DIMENSION  NO(100),SP(100)
       DIMENSION  INODE(150),JNODE(150),ARCOST(150)
       DIMENSION  ISPATH(50)
         COMMON / INTS / CASE,NTYPE,N,M,NARC,ISTART,LAST,PARAM,
     *                   NUMP,UNITF,NVEH,NCOL,NROW,ROWSP,MINE,
     *                   NWD,NDP,NFORWD
         COMMON / INTA / TAIL,HEAD,TIME,FLOW,LINK,LARC,NO,SP,
     *                   INODE,JNODE,ISPATH
         COMMON / REALS/ BIG,TPATH,XLEN,PATHL,FORWD,FORDP,PRT
         COMMON / REALA/ WIDTH,DIST,RESM
       OPEN (UNIT = 10, FILE = 'TIME')
       OPEN (UNIT = 20, FILE = 'DIST')

*_____________________________________________________________________

* (A). PRINT THE RESULTS FOR MINIMUM TIME PATH.
*_____________________________________________________________________

          IF (PARAM .EQ. 1) THEN
               WRITE( 6,711)
               WRITE(10,711)
711            FORMAT(5X,35('-'),/,5X,'MIN  TIME  PATH',/,5X,15('-'))
               IF (CASE .EQ. 1) THEN
                    WRITE( 6,712) NROW,NCOL,ROWSP
                    WRITE(10,712) NROW,NCOL,ROWSP
712                 FORMAT(' ',4X,'VEHICLE UNIT ',/,5X,'FORMATION :',I3,
     *                   ' ROWS',I3,' COLUMNS',/,17X,'ROW SPACE',
     *                   I3,' METERS ')
               ELSE IF (CASE .EQ. 2) THEN
*                   NWD = FORWD * 1000
*                   NDP = FORDP * 1000
                    IF (UNITF .EQ. 1) THEN
```

```
                WRITE( 6,713) NROW,NCOL,NWD,NDP
                WRITE(10,713) NROW,NCOL,NWD,NDP
713             FORMAT(' ',4X,'DISMOUNTED TROOPS',/,5X,
     *            'FORMATION :',' ROWS ',I5,', COLUMNS',I5,/,
     *            16X,' WIDTH',I5,',   DEPTH',I5,' METERS')
                ELSE IF (UNITF .EQ. 2) THEN
                WRITE( 6,714) NWD,NDP
                WRITE(10,714) NWD,NDP
714             FORMAT(' ',4X,'DISMOUNTED TROOPS',/,5X,
     *            'FORMATION :',' WIDTH',I5,',  DEPTH',I6,' METERS')
                END IF
            END IF
                IF (RESM .EQ. 'Y') THEN
                     WRITE( 6,721) MINE
                     WRITE(10,721) MINE
                    FORMAT(/,5X,'1 MINE FIELD ( ARC NUMBER :',I3,')',
721  *                      /,5X,35('-'))
                ELSE IF (RESM .EQ. 'N') THEN
                     WRITE( 6,722)
                     WRITE(10,722)
722                  FORMAT(/,5X,'NO MINE FIELD ARC',/,5X,35('-'))
                END IF

*  WHEN PRT EQUALS 0.0, THERE IS A FEASIBLE ROUTE TO GO.
            IF (PRT .EQ. 0.0) THEN
                IY = 100
                ISUMT = XLEN / 100
                NXLEN = XLEN
                MSUMT = ( MOD( NXLEN, IY ) * 60 ) / 100
            WRITE( 6,731) ISUMT, MSUMT
            WRITE(10,731) ISUMT, MSUMT
731         FORMAT(' ',/,5X,'SUM OF TRAVERSAL TIME : ',
     *                  I3,2X,'HOUR',I5,2X,'MINUTE')
                ITHR   = TPATH / 100
                NTPATH = TPATH
                MINUT  = ( MOD( NTPATH, IY ) * 60 ) / 100
            WRITE( 6,732) ITHR, MINUT
            WRITE(10,732) ITHR, MINUT
732         FORMAT(' ',/,5X,'TOTAL TRAVERSAL TIME  : ',
     *                  I3,2X,'HOUR',I5,2X,'MINUTE')
            WRITE( 6,733)  PATHL
            WRITE(10,733)  PATHL
733         FORMAT(' ',/,5X,'TOTAL PATH  LENGTH    : ',
     *                  F7.2,2X,'KM')
            WRITE( 6,734)
            WRITE(10,734)
734         FORMAT(' ',/,5X,'NODE NUMBER ALONG MIN TIME PATH ')
            WRITE( 6,735)(I,ISPATH(I), I =1, NUMP)
            WRITE(10,735)(I,ISPATH(I), I =1, NUMP)
735         FORMAT(10X,I3,3X,I3)
            WRITE( 6,736)
            WRITE(10,736)
736         FORMAT(' ',/,5X,' ARC NUMBER ALONG MIN TIME PATH ')
            WRITE( 6,750)(I,LARC(I), I =1, NUMP-1)
            WRITE(10,750)(I,LARC(I), I =1, NUMP-1)
750         FORMAT(10X,I3,3X,I3)
            WRITE(6,760) XLEN
760         FORMAT(' ',/,5X,'TOTAL NUMBER ALONG MIN TIME PATH :',
     *                  F5.1)

*  WHEN PRT EQUALS 1.0, THERE IS NO FEASIBLE ROUTE TO GO.
            ELSE IF (PRT .EQ. 1.0) THEN
                     WRITE( 6,741)
                     WRITE(10,741)
                   FORMAT(/,5X,'THERE IS NO FEASIBLE ROUTE TO GO.')
            END  IF
*-------------------------------------------------------------------
* (B). PRINT THE RESULTS FOR MINIMUM DISTANCE PATH.
*-------------------------------------------------------------------
```

```
      ELSE IF (PARAM .EQ. 2) THEN
           WRITE( 6,811)
           WRITE(20,811)
811        FORMAT(5X,35('-'),/,5X,'MIN  DISTANCE  PATH',/,5X,19('-'))
           IF (RESM .EQ. 'Y') THEN
               WRITE( 6,821) MINE
               WRITE(20,821) MINE
821            FORMAT(/,5X,'1 MINE FIELD ( ARC NUMBER :',I3,')',
     *                    /,5X,35('-'))
           ELSE IF (RESM .EQ. 'N') THEN
               WRITE( 6,822)
               WRITE(20,822)
822            FORMAT(/,5X,'NO MINE FIELD ARC',/,5X,35('-'))
           END IF
           WRITE( 6,831)
           WRITE(20,831)
831        FORMAT(' ',/,5X,'NODE NUMBER ALONG MIN DISTANCE PATH ')
           WRITE( 6,832)(I,ISPATH(I), I =1, NUMP)
           WRITE(20,832)(I,ISPATH(I), I =1, NUMP)
832        FORMAT(10X,I3,3X,I3)
           WRITE( 6,833)
           WRITE(20,833)
833        FORMAT(' ',/,5X,' ARC NUMBER ALONG MIN DISTANCE PATH ')
           WRITE( 6,834)(I,LARC(I), I =1, NUMP-1)
           WRITE(20,834)(I,LARC(I), I =1, NUMP-1)
834        FORMAT(10X,I3,3X,I3)
           WRITE( 6,860) XLEN
           WRITE(20,860) XLEN
860        FORMAT(' ',/,5X,'TOTAL DISTANCE ALONG MIN DIST PATH :',
     *                    F5.1,' KM')
      END  IF

* PRINT THE INPUT DATA FOR CHECKING.
           WRITE(6,921)
           WRITE(6,920) (NO(I),TAIL(I),HEAD(I),TIME(I),FLOW(I),
     *                   DIST(I),WIDTH(I),SP(I),I=1,NARC)
                IF (PARAM .EQ. 1) THEN
                    MU = 10
                ELSE IF (PARAM .EQ. 2) THEN
                    MU = 20
                END  IF
           WRITE(MU,921)
           WRITE(MU,920) (NO(I),TAIL(I),HEAD(I),TIME(I),FLOW(I),
     *                   DIST(I),WIDTH(I),SP(I),I=1,NARC)
921        FORMAT(' ',//,' ',' NO  TAIL  HEAD  TIME  FLOW  DIST',
     *                  ' WIDTH   SPEED')
920        FORMAT(2X,I3,2I5,2I6,3X,F4.1,3X,F6.3,2X,I4)
      RETURN
      END
```

APPENDIX E
COMPUTER EXEC PROGRAM

This appendix contains two computer exec programs used to support the algorithms in Chapter III and IV.

SINGLE ARC EXEC PROGRAM

This exec program is used for the computer program for single arc attributes.

```
&TRACE OFF
&TYPE Please provide the FILENAME for your VS FORTRAN program.
&READ VAR &FN
&TYPE Do you need to compile your program ? (Y)
&READ VAR &R_COMPILE
&IF &R_COMPILE NE Y &GOTO -RUN
-H FORTVS &FN
&IF &RC EQ 0 &SKIP 9
&TYPE Your program did not compile; check for errors.
&TYPE Do you wish to XEDIT the program file? (Y)
&READ VAR &RESP1
&IF &RESP1 NE Y &EXIT 1
&COMMAND XEDIT &FN FORTRAN A
&TYPE Do you wish to run the program again? (Y)
&READ VAR &RESP2
&IF &RESP2 EQ Y &GOTO -H
&EXIT 1
-RUN &TYPE Do you wish your INPUT to be from the terminal? (Y)
&READ VAR &IN
&IF &IN NE Y &GOTO -RUN2
-INPUT_FILE FILEDEF 01 DISK NB71    DATA A1
-INPUT_FILE FILEDEF 02 DISK NBNOD1  DATA A1
-INPUT_FILE FILEDEF 03 DISK NBARC1  DATA A1
-INPUT_FILE FILEDEF 04 DISK SPEED   DATA A1
FILEDEF 05 TERMINAL
-RUN2 &TYPE Do you wish your OUTPUT to go to the terminal? (Y)
&READ VAR &OUT
&IF &OUT NE Y &GOTO -OUTPUT_FILE
&GOTO -LOAD
-OUTPUT_FILE FILEDEF 06 DISK &FN OUTPUT A (LRECL 133
-LOAD LOAD &FN (START
&IF &RC EQ 0 &SKIP 9
&TYPE Your program did not run correctly; check for errors.
&TYPE Do you wish to XEDIT the program file? (Y)
&READ VAR &RESP3
&IF &RESP3 NE Y &EXIT 2
&COMMAND XEDIT &FN FORTRAN A
&TYPE Do you wish to run the program again? (Y)
&READ VAR &RESP4
&IF &RESP4 EQ Y &GOTO -H
&EXIT 2
&IF &OUT EQ Y &GOTO -REDO
&TYPE Your output is in the file &FN OUTPUT A
&TYPE Do you wish  to BROWSE your output? (Y)
&READ VAR &RESP
&IF &RESP EQ Y &COMMAND BROWSE &FN OUTPUT A
&TYPE Print your output file?  (Y)
&READ VAR &RESP7
&IF &RESP7 EQ Y &COMMAND PRINT &FN OUTPUT A
-REDO
&TYPE Do you wish to XEDIT the program file? (Y/N)
```

```
&READ VAR &RESP5
&IF &RESP5 EQ Y XEDIT &FN FORTRAN A
&TYPE Do you wish to run the program again? (Y)
&READ VAR &RESP6
&IF &RESP6 EQ Y AND &RESP5 EQ Y &GOTO -H
&IF &RESP6 EQ Y &GOTO -RUN
&EXIT
```

NETWORK EXEC PROGRAM

This exec program is used for computer program for Cartesian space network.

```
&TRACE OFF
&TYPE Please provide the FILENAME for your VS FORTRAN program.
&READ VAR &FN
&TYPE Do you need to compile your program ? (Y)
&READ VAR &R_COMPILE
&IF &R_COMPILE NE Y &GOTO -RUN
-H FORTVS &FN
&IF &RC EQ 0 &SKIP 9
&TYPE Your program did not compile; check for errors.
&TYPE Do you wish to XEDIT the program file? (Y)
&READ VAR &RESP1
&IF &RESP1 NE Y &EXIT 1
&COMMAND XEDIT &FN FORTRAN A
&TYPE Do you wish to run the program again? (Y)
&READ VAR &RESP2
&IF &RESP2 EQ Y &GOTO -H
&EXIT 1
-RUN &TYPE Do you wish your INPUT to be from the terminal? (Y)
&READ VAR &IN
&IF &IN NE Y &GOTO -RUN2
-INPUT_FILE FILEDEF 01 DISK NETA    DATA A1
-INPUT_FILE FILEDEF 02 DISK NETB    DATA A1
FILEDEF 05 TERMINAL
-RUN2 &TYPE Do you wish your OUTPUT to go to the terminal? (Y)
&READ VAR &OUT
&IF &OUT NE Y &GOTO -OUTPUT_FILE
&GOTO -LOAD
-OUTPUT_FILE FILEDEF 06 DISK &FN OUTPUT A (LRECL 133
-LOAD LOAD &FN (START
&IF &RC EQ 0 &SKIP 9
&TYPE Your program did not run correctly; check for errors.
&TYPE Do you wish to XEDIT the program file? (Y)
&READ VAR &RESP3
&IF &RESP3 NE Y &EXIT 2
&COMMAND XEDIT &FN FORTRAN A
&TYPE Do you wish to run the program again? (Y)
&READ VAR &RESP4
&IF &RESP4 EQ Y &GOTO -H
&EXIT 2
&IF &OUT EQ Y &GOTO -REDO
&TYPE Your output is in the file &FN OUTPUT A
&TYPE Do you wish  to BROWSE your output? (Y)
&READ VAR &RESP
&IF &RESP EQ Y &COMMAND BROWSE &FN OUTPUT A
&TYPE Print your output file?  (Y)
&READ VAR &RESP7
&IF &RESP7 EQ Y &COMMAND PRINT &FN OUTPUT A
-REDO
&TYPE Do you wish to XEDIT the program file? (Y/N)
&READ VAR &RESP5
&IF &RESP5 EQ Y XEDIT &FN FORTRAN A
&TYPE Do you wish to run the program again? (Y)
&READ VAR &RESP6
&IF &RESP6 EQ Y AND &RESP5 EQ Y &GOTO -H
&IF &RESP6 EQ Y &GOTO -RUN
&EXIT
```

APPENDIX F
UNIT TYPE AND FORMATION

Table 18 shows the integer value and the unit type it represents.

TABLE 18
TYPE OF UNIT

Integer Value	Type of Unit
1	Tracked Vehicle
2	Wheeled Vehicle
3	Dismounted Troops

Table 19 shows the integer value and the unit formation it represents.

TABLE 19
UNIT FORMATION

Integer Value	Unit formation
1	Deployment (multiple column)
2	Undeployment (single column)

APPENDIX G
RATING CONE INDEX

Figure G.1 shows an example of how the rating cone index can be used [Ref. 4: p. 20].

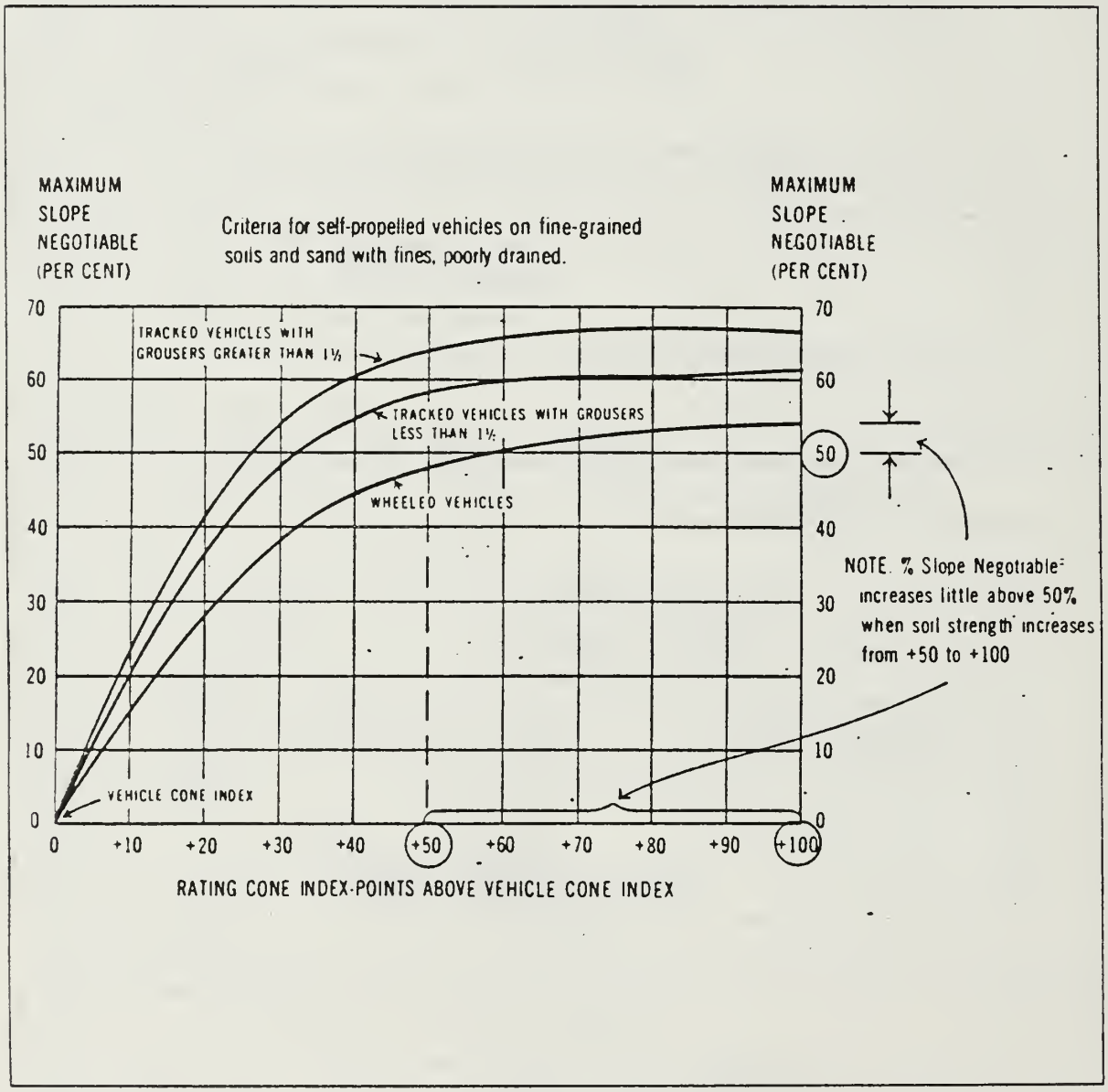

Figure G.1 Rating Cone Index.

LIST OF REFERENCES

1. Krupenevich, Thomas P., *Network Representation for Combat Models*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, December 1984.

2. Needles, Christopher J., *Parameterization of Terrain in Army Combat Analysis*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, March 1976.

3. Thomas, George B., Finney, Ross L., *Calculus and Analytic Geometry*, Addison-Wesley Publishing Co., May 1984.

4. Department of the Army, Field Manual 21-33, *Terrain Analysis*, Washington, D.C., May 1978.

5. Fletcher, Douglas L., *Models for Avenue of Approach Generation and Planning Process for Ground Combat Forces*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1986.

6. Craig, Dean E., *A Model for The Planning of Maneuver Unit and Engineer Asset Placement*, M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1985.

7. Jensen, Paul A., Barnes, J. Wesley, *Network Flow Programming*, John Wiley & Sons, Inc., 1980.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Deputy Undersecretary of the Army for Operations Research Room 2E261, Pentagon Washington, D.C. 20310	1
4.	Director U.S. Army TRADOC Analysis Center White Sands Missile Range New Mexico 88002	1
5.	Commander U.S. Army TRADOC Analysis Center Attn: Mr. Reed Davis Fort Leavenworth, KS 66027	2
6.	Director Attn: Mr. E. B. Vandivar III U.S. Army Concepts Analysis Agency Bethesda, MD 20814	1
7.	Bell Hall Library U.S. Army Combined Arms Center Fort Leavenworth, KS 66027	1
8.	Dr. Samuel H. Parry, Code 55Py Department of Operations Research Naval Postgraduate School Monterey, CA 93943	5
9.	Dr. Arthur L. Schoenstadt, Code 53Zh Department of Mathematics Naval Postgraduate School Monterey, CA 93943	2
10.	Dr. Peter Purdue, Code 55 Department of Operations Research Naval Postgraduate School Monterey, CA 93943	1

11.	MAJ Bard Mansager, Code 55 Department of Operations Research Naval Postgraduate School Monterey, CA 93943	1
12.	Library, P.O. Box 77 Postal Code 130-09 Gong Neung Dong, Dobong Gu Seoul, Republic of Korea	1
13.	MAJ Choi, Seok Cheol Postal Code 601-00 Gaegeum 2 Dong, 566-35 (9 Tong 6 Ban) Busan-Jin-Gu, Busan, Republic of Korea	2
14.	Library, P.O. Box 2 Postal Code 140-01 Yongsan Gu, Yongsan Dong Seoul, Republic of Korea	1
15.	LTC Jung, Yun Su 1192 Del Monte APT #A Monterey, CA 93943	1
16.	Director Attn: COL Tony Brinkley Studies and Analysis Directorate Headquarters, U.S. Army TRADOC Fort Monroe, VA 23651	1
17.	Department of Operations Sciences Attn: MAJ Dan Reyen AFIT / ENS Wright Patterson AFB, OH 45433	1
18.	Director U.S. Army Models Management Office Combined Arms Center Fort Leavenworth, KS 66027	1
19.	MAJ Kang, Sun Mo Postal Code 130-60 Kyung Gi Do, Yang Pyung Goon Yang Pyung Eub, Obin Ri, 233-11 Seoul, Republic of Korea	1
20.	MAJ Kim, Dae Sik 1198 8th Street APT #2 Monterey, CA 93943	1
21.	MAJ Yoon, Sang Il 1287 Playo Avenue APT #B Seaside, CA 93955	1

Thesis
C448857 Choi
c.1 Determination of network attributes from a high resolution terrain data base.